Naval Research Laboratory

Washington, DC 20375-5320

NRL/MR/7230--04-8760

NRL Atmospheric Correction Algorithms for Oceans: *Tafkaa* User's Guide

MARCOS J. MONTES
BO-CAI GAO

Coastal and Ocean Remote Sensing Branch
Remote Sensing Division

CURTISS O. DAVIS

Remote Sensing Division

March 22, 2004

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
March 22, 2004	Memorandum	January 2002-December 2003

4. TITLE AND SUBTITLE

NRL Atmospheric Correction Algorithms for Oceans: *Tafkaa* User's Guide

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61153N

6. AUTHOR(S)

Marcos J. Montes, Bo-Cai Gao, and Curtiss O. Davis

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory, Code 7230
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/7230--04-8760

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5660

10. SPONSOR / MONITOR'S ACRONYM(S)

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Tafkaa is an atmospheric correction algorithm for remote sensing of ocean color that has been developed at the Naval Research Laboratory (NRL). This document describes how to use *Tafkaa*, and presents some background on *Tafkaa*, including differences and similarities in both the current tabular and 6S versions. Descriptions of various options, restrictions, assumptions, keywords, input files, and output files associated with *Tafkaa* are also presented.

15. SUBJECT TERMS

Atmospheric correction; Hyperspectral remote sensing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Marcos J. Montes
Unclassified	Unclassified	Unclassified	UL	40	19b. TELEPHONE NUMBER *(include area code)* (202) 767-7308

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Contents

1 Introduction & Background

Read the entire guide before running *Tafkaa*. It will answer many questions, and make subsequent use of *Tafkaa* more efficient and productive.

Items listed in a `teletype` font represent keywords or values you might see in an image header file (§6.3.1) or a *Tafkaa* input file (§6.3.2).

Tafkaa is an algorithm for atmospheric correction of imaging spectrometry data. It is not one, but two separate executables. `tafkaa_6s` is based on ATmospheric REMoval (ATREM) 4.0, and uses Second Simulation of the Satellite Signal in the Solar Spectrum (6S) for it's scattering calculations. It cannot correct for the specular reflection of the air-water interface. `tafkaa_tabular` uses pre-calculated lookup-tables for the various scattering quantities; the lookup tables include the specular effects of the air-water interface. As described above, portions of the algorithms are different. The portions of the algorithms that are identical use the same source files. However, most of the algorithmic development and focus has been in the `tafkaa_tabular` path. Therefore, reference to *Tafkaa* should be read as applying to `tafkaa_tabular`, unless otherwise noted.

The tabular version of *Tafkaa* is currently configured for hyperspectral observation from both aircraft and spacecraft viewing a sea-level surface. The current 6S version of *Tafkaa* can deal with aircraft in the atmosphere, and with elevated features; however, as it cannot account for specular reflection from the ocean's surface, it is primarily for use over the land.

Development and discussion of the equations below may be found in Fraser et al. [1997], Vermote et al. [1994, 1997] and Gao et al. [1993]. Applications and implementations of *Tafkaa* may be found in Gao et al. [2000] and Montes et al. [2001, 2003].

Our basic equation may be written as[1]

$$\begin{aligned} L_{\rm t} &= L_0 + L_{\rm sfc} t + L_{\rm g} t' \ , \qquad (1) \\ \text{where} & \\ L_0 &= L_0(\lambda;\theta,\phi;\theta_0,\phi_0;\tau_{\rm a};z_{\rm sen},z_{\rm sur}) \ ; \\ L_{\rm sfc} &= L_{\rm sfc}(\lambda;\theta,\phi;\theta_0,\phi_0;W;\tau_{\rm a};z_{\rm sur}) \ ; \\ L_{\rm g} &= L_{\rm g}(\lambda;\theta,\phi;\theta_0,\phi_0;W;C;\tau_{\rm a};z_{\rm sur}) \ ; \\ t &= t(\lambda;\theta;\tau_{\rm a};z_{\rm sen},z_{\rm sur}) \ ; \\ t' &= t'(\lambda;\theta;\tau_{\rm a};z_{\rm sen},z_{\rm sur}) \ . \end{aligned}$$

L_0 is the so-called "path radiance" term; that is, sunlight scattered by the atmosphere that never interacts with the ground. $L_{\rm sfc}$ accounts for the specular reflections off the target, and is only necessary over water; otherwise it is 0. $L_{\rm g}$ is the radiance reflected (in a Lambertian manner) from the ground (or water); $L_{\rm w} = L_{\rm g}$ for observations over water.

In the discussion below, it will be convenient to combine the first two terms on the right hand side of Equation 1 so that

$$L_{\rm sa} = L_0 + L_{\rm sfc} t \ . \qquad (2)$$

We assume that the absorptive processes are multiplicative and can be factored out directly; in essence we are breaking the transmission into absorptive and scattering parts. This is valid as long as either the absorptive or scattering processes dominate. There are some errors if they are similar in magnitude. This should only be a problem when the absorptive processes are changing rapidly (from the scattering dominated regime to the absorptive dominated regime) such as near edges of strong absorptive features. Thus,

$$L_{\rm t} = T_{\rm g} \left(L'_{\rm sa} + L_{\rm g} t'_{\rm u} \right) \ , \qquad (3)$$

Manuscript approved March 5, 2004.

[1] All the quantities in this document are described in the glossary section on page 20.

where T_g is the transmission due to the absorptive processes in the gas.

We would like to work with reflectances instead of radiances, thus we divide by the incident flux at the top of the atmosphere $\mu_0 E_0$, and multiply by π:

$$\frac{\pi L_t}{\mu_0 E_0} = T_g \left[\frac{\pi L'_{sa}}{\mu_0 E_0} + \frac{\pi L_g t'_u t_d}{\mu_0 E_0 t_d} \right] , \tag{4}$$

where we have multiplied the numerator and denominator of the second term on the right by the downward transmittance t_d in order to obtain the ground (or water-leaving) reflectance.

We will use the reflectances defined as

$$\rho^*_{obs} \equiv \frac{\pi L_t}{\mu_0 E_0} , \tag{5}$$

$$\rho^*_{atm+sfc} \equiv \frac{\pi L'_{sa}}{\mu_0 E_0} , \text{ and} \tag{6}$$

$$\rho_g \equiv \frac{\pi L_g}{\mu_0 E_0 t_d} . \tag{7}$$

A final step is to remove the effect of multiple reflections of the ground or water leaving radiance from t'_u. Neglecting specular reflection of the water leaving radiance from the ocean surface and assuming Lambertian reflectance for the water leaving radiance, we find

$$t'_u = \frac{t_u}{1 - \bar{s}\rho_g} , \tag{8}$$

where $\bar{s} = \bar{s}(\lambda, \tau_a)$ is the average reflectivity of the atmosphere when illuminated by a Lambertian source at its base. This implies that

$$\rho^*_{obs} = T_g \left[\rho^*_{atm+sfc} + \frac{\rho_g t_u t_d}{1 - \bar{s}\rho_g} \right] . \tag{9}$$

A more exact calculation would also include multiple reflections off of the specular ocean surface (water leaving light that is reflected downward of the atmosphere then specularly reflected by the surface); however, this effectively implies that the water leaving radiance is not Lambertian, and this makes it much more difficult to extract ρ_w.

In the above equation, we regard ρ^*_{obs} as the measured quantity (since μ_0 and E_0 are well known); ρ_g is the quantity we wish to derive, and $\rho^*_{atm+sfc}$, $t_u, t_d, \bar{s}$, and T_g are to be calculated by *Tafkaa*. Thus, we rewrite the above equation with ρ_g on the left as

$$\rho_g = \frac{\frac{\rho^*_{obs}}{T_g} - \rho^*_{atm+sfc}}{t_u t_d + \bar{s}\left(\frac{\rho^*_{obs}}{T_g} - \rho^*_{atm+sfc}\right)} . \tag{10}$$

2 Gaseous Absorptive Effects

The gaseous absorptive effects are treated differently for different sets of gases. The plane-parallel, two-way transmission of certain gases is calculated on a high-resolution (spectrally) grid, and this is carefully convolved to a medium resolution grid. At this point, ozone (O_3) and nitrogen dioxide (NO_2) are included, and finally this is convolved with the instrumental response function to the instrumental grid.

In *ATREM* and previous versions of *Tafkaa* the instrumental response function (IRF) was assumed to be Gaussian, thus only the center wavelength and the full-width at half-maximum (FWHM) were needed in order to perform the convolution. Certain instruments such as AVIRIS are very well represented by this model. Other instruments may not be. In order for *Tafkaa* to be as general as possible, I have added the ability for it to use a user-supplied instrumental response function. The default behavior is still to use the Gaussian IRF. If the keyword `tafkaa_instrumental_response_file` is set, then *Tafkaa* will use that value as the filename that contains the instrumental response function. A full description of the correct format to use for the IRF file is presented in Appendix D.

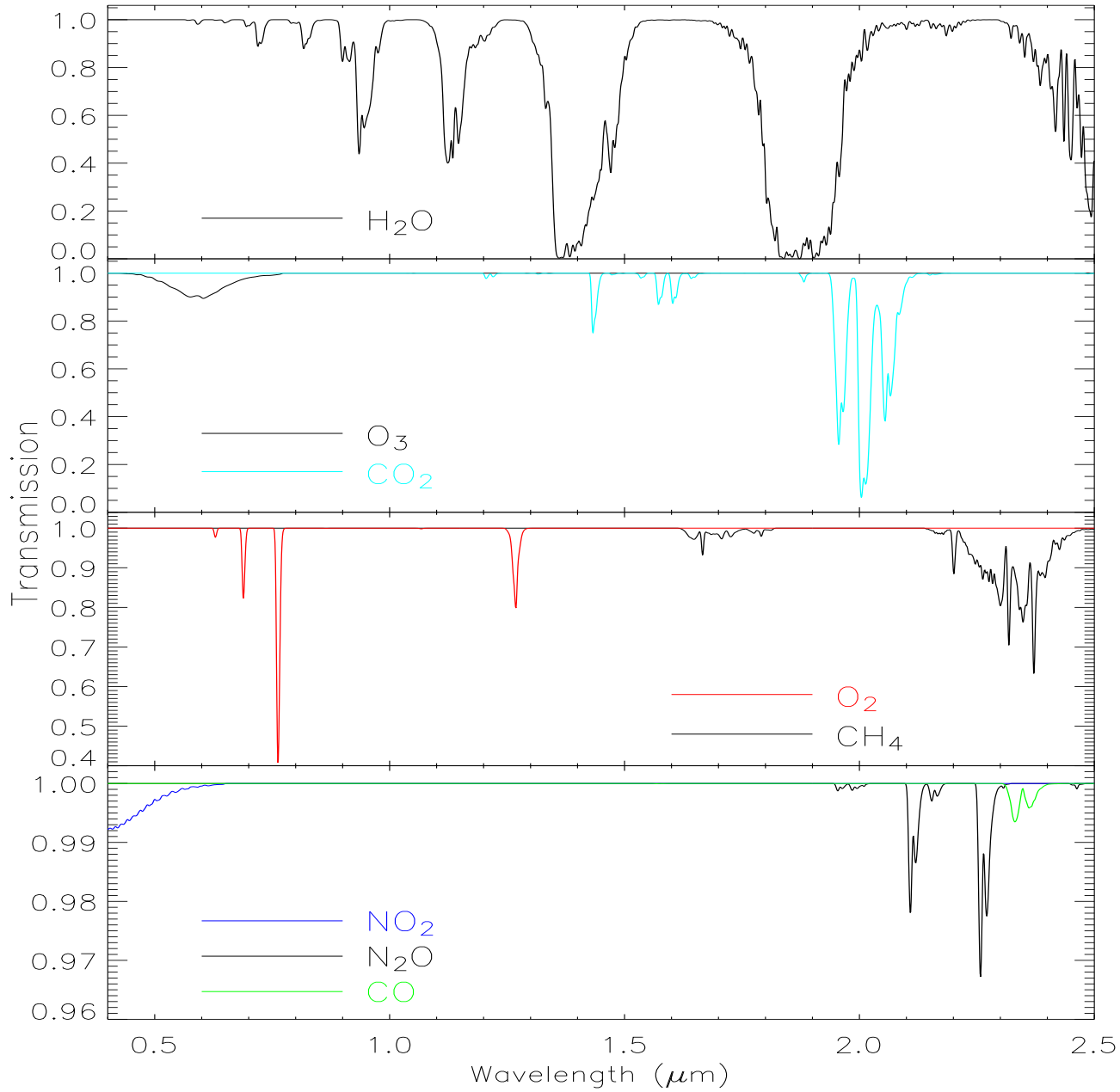

Figure 1: The transmission functions for the gases used in *Tafkaa*, showing the location of the absorption features due to each gas. The path is specified by $\theta_0 = 50°$, $\theta = 0°$, a sea-level surface, and a sensor at the top of the atmosphere. Typical amounts of each gas were used to construct this plot. The legends of each plot indicate the gas in each one. The horizontal scales of each plot are identical; the vertical scales are different. These figures were prepared with $\Delta\lambda = 1$ nm and FWHM = 5 nm.

Temperature and pressure dependent effects are computed [Ridgway, personal communication, 1996] to generate a database of gaseous absorption coefficients at a spectral resolution of 0.05 cm^{-1} in the 0.56 – 3.1 μm range for H_2O, CO_2, N_2O, CO, CH_4, and O_2. These files are stored in the *Tafkaa* data directory, and each consists of 300,000 spectral locations and 19 atmospheric layers.

Water vapor has a unique treatment. *Tafkaa* (and its predecessor *ATREM*) as originally written calculated the water vapor amount on a pixel-to-pixel basis on the fly. In order to do this, 60 different water vapor values are used to calculate 60 high resolution gas transmission spectra, creating a water-vapor lookup table for the particular view & solar geometries and surface & sensor altitudes. Indeed, this is the most time

consuming portion of *Tafkaa* before the pixel-to-pixel processing begins, as this implies the high-to-medium resolution convolution needs to be done sixty times. Alternately, one can now neglect water vapor, or include a particular amount of water vapor. Either of these options greatly decreases the startup time, but water vapor features are not corrected at all (or as well). Still, if the strong water vapor bands are not available or the signal to noise in the bands is poor, these new features allow one to continue the analysis with *Tafkaa*.

As long as water vapor is selected, the water vapor volume mixing ratios of the selected atmospheric model are used to place the relative amounts of water vapor in the different layers of the atmosphere. Thus, the portions of the spectrum most affected by the choice of atmospheric model are those portions containing water vapor features.

Ozone absorption coefficients in the $0.3 - 0.8\,\mu$m region were derived from `LOWTRAN7` [Kneizys et al., 1988] O_3 transmittance spectra calculated with the US76 atmospheric model. The wavelength spacing between any two adjacent points is 0.1 nm, with a resolution of 0.2 nm. *Tafkaa* assumes that O_3 occurs in a nearly infinitesimally thin layer at an altitude of 27 km above mean sea level. As long as the sensor is well below or above the true ozone layer, this is not a problem. If the sensor is in the true ozone layer, *Tafkaa* will probably return incorrect results in this portion of the spectrum. The absorption effects of man-made ozone are usually quite small compared to those of the stratospheric ozone, so the errors in neglecting the low-altitude man-made distribution are quite small.

The NO_2 cross sections in the $0.3 - 0.8\,\mu$m region correspond to the curve in the top plot of Fig. 1 of Solomon et al. [1999]. Adjacent points are spaced at 0.1 nm intervals, each with a resolution of 0.2 nm. A column amount of $5{\times}10^{15}$ molecules reduces the transmittance at 410 nm by $\sim 0.6\%$, and currently it is treated as having the same distribution as described above for ozone. The keyword `tafkaa_atmo_no2_scale` specifies a multiplicative factor to be used to scale the amount of NO_2 present. NO_2 is also produced by burning fossil fuels, and so is present in the lower layers of the atmosphere. In fact, in certain locations there may be more present in the troposphere due to pollution than is present in the stratosphere (see, for example, `http://www.iup.physik.uni-bremen.de:8083/doas/no2_from_scia.htm`). In these cases, results from *Tafkaa* will not be as accurate.

Both the ozone amount and the NO_2 scale factor may be adjusted to obtain the correct absorption if the user is particularly careful in choosing the values of each that are given to *Tafkaa*. Both are scaled internally by $\mu_0^{-1} + \mu^{-1}$ when the sensor is above 27 km, and by μ_0^{-1} when the sensor is below 27 km. Since it is only the total amount that matters, it is possible for one to determine a reasonable input if the tropospheric and stratospheric amounts can be measured separately. Until modifications are made to the O_3 and NO_2 distributions of *Tafkaa*, results should be accurate for sensors that are not in the real ozone layer. Sensors in the ozone layer need to have the ozone amounts scaled carefully (as suggested above) in order to remove the effects of ozone. Results for airborne sensors may be less accurate in polluted areas where significant amounts of NO_2 are present below the sensor. Assuming that there is $5{\times}10^{15}$ molecules of NO_2 in both the stratosphere and in the troposphere below the sensor, and assuming $\mu_0 = \mu = 1$, and that the user specified an amount of only $5{\times}10^{15}$ molecules, the transmission calculated by *Tafkaa* would be too high by $\sim 0.6\%$ at ~ 410 nm. This will lead to an underestimate of $\rho_{\rm g}$, but the exact amount is difficult to determine without a greater knowledge of the of the other quantities in Eq. 10. However, given a tropospheric aerosol associated with the increased NO_2, and $\tau_{\rm aer}(0.55\,\mu{\rm m}) = 0.2$, $\Delta\rho_{\rm w} \sim -0.0016$ is to be expected; that is, more is subtracted than should otherwise have been subtracted. Regions with low pollution will not have this concern.

The portion of the program that calculates the gaseous absorptive effects will run somewhat faster if fewer gases are selected. Selection should be made based on whether or not there is absorption due to each gas in the instrument's spectral range (see Fig. 1 or Gao et al. [1993] for plots of transmittance vs. λ for different gases that are included with *Tafkaa*). In particular, for sensors with a spectral response $\lesssim 1\,\mu$m, one needs to include only NO_2, O_3, H_2O, and O_2.

3 Application to `tafkaa_6s`

Thorough discussions of the *6S* version of *Tafkaa* (i.e., the one we call `tafkaa_6s`) can be found in Gao et al. [1993] and Gao et al. [1997].

`tafkaa_6s` uses a modified implementation of the subroutine *6S* [Vermote et al., 1997] to calculate the quantities $\rho_{\rm atm}, t_{\rm u}, t_{\rm d}$, and $\bar{s}$. The aerosol types are listed in Appendix H.

6S is able to read files in order to use other aerosol models. This is disabled in the modified implementation used in `tafkaa_6s`.

The modified implementation of *6S* provides $\rho_{\rm atm}$, not $\rho^*_{\rm atm+sfc}$. Specular reflection from the ocean surface of both the direct solar beam and of the skylight is not accounted for. In "nice" conditions (favorable observing and solar geometry and little or no wind), the corrections provided by `tafkaa_6s` may have a reasonable shape (but > 0 in the Near Infrared (NIR)) over the water, but the magnitudes will be wrong. Mobley [1999], for example, discusses the importance specular reflection from an ocean surface.

4 Application to `tafkaa_tabular`

The principal difference between the tabular version of *Tafkaa* (i.e., `tafkaa_tabular`) and `tafkaa_6s` is the source of the quantities $\rho^*_{\rm atm+sfc}, t_{\rm u}, t_{\rm d}$, and $\bar{s}$. A computer program by Zia Ahmad (a somewhat modified version of that discussed in Ahmad and Fraser [1982]) performs vector radiative transfer computations in order to calculate the desired quantities. The calculations are performed for a variety of solar and observational geometries, as well as for five general aerosol types[2] (each at five relative humidities and ten optical depths). Furthermore, the quantities were calculated at only a few specially selected wavelengths in atmospheric windows where the molecular (gas) absorption is negligible. The geometrical, wavelength, optical depth, and relative humidity grids may be found in Appendix C.

The tables were constructed precisely because it takes a very long time to calculate the aforementioned quantities, and this led to careful choices of the aerosol models, relative humidities, observational and solar angles, and optical depths where the calculations would be performed. In practice, linear interpolations are performed in the angular quantities in order to reduce the size of the table that needs to be searched in order to determine best-fit aerosol model. Finally, since the ocean surface is not completely flat (due to the wind), and the sky is not dark (due to atmospherically scattered light), it is necessary to calculate the above quantities over a realistic wind-blown ocean surface ($W = 2$ m/s, 6 m/s, and 10 m/s). The wind speed is used solely for the properties of the lower boundary condition and is **not** used to calculate aerosol properties in any way, as it is for some aerosol models, such as those contained in *MODTRAN*.

We have also calculated $\rho_{\rm atm}$ for a zero-reflectance Lambertian and assembled these quantities into the proper format for `tafkaa_tabular`, which allows us to use the same aerosol models over both the ocean and land.

The total number of tables currently needed for sea-level uses of `tafkaa_tabular` from above the atmosphere is 7: one each table of $t_{\rm u}, t_{\rm d}$, and $\bar{s}$; a table of $\rho^*_{\rm atm+sfc}$ for each of three velocities, appropriate for use over water; and a table of $\rho_{\rm atm}$ appropriate for use over land.

`tafkaa_tabular` uses tables for several values of $z_{\rm sen}$. Any altitude above 84 km is considered to be "above the atmosphere" and all calculations at or above this altitude will use the same table entries. The additional tables at each $z_{\rm sen}$ are $t_{\rm u}$, $\rho^*_{\rm atm+sfc}$ over water at each of the three wind speeds and $\rho_{\rm atm}$. The list of $z_{\rm sen}$ the tables were computed at is in Appendix C. For any other altitude $\rho^*_{\rm atm+sfc}$, $\rho_{\rm atm}$, and $t_{\rm u}$ are calculated via interpolation in pressure.

4.1 Determining aerosol parameters

In order to determine the correct aerosol parameters, we assume that there is no water leaving radiance at particular wavelengths. Due to the strong absorption of light by water beyond about 0.75 μm (see, for example, Kou et al. [1993], as well as the compendium at `http://omlc.ogi.edu/spectra/water/abs/`), this is true for most waters. (There may, however, be significant $L_{\rm w}$ out to $\sim 1\,\mu$m in waters with high amounts of backscattering caused by a large sediment load.) In the case where $L_{\rm w} = 0$, then the numerator of Eq. 10 must be 0. At the wavelengths chosen the observed reflectance ($\rho^*_{\rm obs}$) is equal to the atmospheric reflectance ($\rho^*_{\rm atm+sfc}/T_{\rm g}$). Thus, the aerosol type and optical depth are determined by matching the observed atmospheric reflectance to the calculated atmospheric reflectance at two or more wavelengths, and using a least squares determination in order to determine the best match. (Linear interpolations are performed on

[2]The aerosols used are described in more detail in Appendix H.

the aerosol optical depth[3], but not on relative humidity or between the general model types.) It is frequently necessary to use only the wavelengths $> 1\,\mu$m when the water is very turbid. [Or, more exactly, if in the upper few optical depths (say, $z < 2/a_{\mathrm{w}}$) the light at the wavelengths in question is subject to much greater scattering so that there is measurable water leaving radiance. Thus the water is not a dark surface, violating one of our assumptions.] Once the parameters of the best-fit aerosol model are determined, we can use the appropriate $\rho^*_{\mathrm{atm+sfc}}, t_{\mathrm{u}}, t_{\mathrm{d}}$, and $\bar{s}$ in order to atmospherically correct the observed reflectance, and thus determine the water leaving radiance.

Based on the above discussion, it is obvious why we cannot determine the aerosol type and optical depth over the land with this method: the land is rarely dark enough at any particular wavelength in order for the assumption "the observed reflectance equals the atmospheric reflectance," to be true.

4.1.1 Aerosol properties and measurement error

Both measurement and discretization error come into play in the determination of aerosol properties. This is largely because the reflectance is fairly small in the portions of the spectrum that we must use; thus discretization errors are fractionally larger than they might otherwise be. Monte Carlo style numerical experiments show that in order to obtain the correct results, it is best to average several pixels, possibly as few as 20 (with about 3% measurement errors, discretization errors similar to those of AVIRIS, and using the reflectance at four wavelengths to determine the aerosol properties). This is not currently implemented in `tafkaa_tabular`. Also note that if discretization errors are larger than the separation between models or larger than the measurement errors, averaging can't help.

5 Data Issues

5.1 `tafkaa_tabular` processing with only *VNIR*

If only the *VNIR* bands ($\lambda < 1\,\mu$m) are available, then aerosol determination needs to use the $0.75\,\mu$m and $0.865\,\mu$m bands.

- The first option will be to use just the $0.75\,\mu$m and $0.865\,\mu$m bands – the minimum needed in order to determine both the optical depth and aerosol type (assuming no binning of pixels). This should work reasonably well in areas where there is no silt or other material suspended in the topmost part of the ocean, that is, in truly open ocean situations.
- The second option is to use other available information in order to determine aerosol types. This may be possible if there are measurements of the aerosol properties (near enough in time and location) available.
- A third option is to determine the aerosol type and optical depth from a nearby (in time and location) scene where both *VNIR* and *SWIR* observations are available.
- Finally, we could use just one band, say the $0.75\,\mu$m band, in order to determine one of the parameters (optical depth or aerosol type) if the other quantity is independently available. *This option is not yet available in the hyperspectral version of Tafkaa, but it has been included in a multispectral version.*

5.2 Differing geometries across a scene

The relative importance of changing geometry in a particular image depends on many parameters. Some instruments, such as Hyperion, have relatively narrow fields of view ($\sim 0.^\circ 5$) such that changing view geometry is not much concern. Other instruments, such as AVIRIS and PHILLS, may be flown with wide fields of view ($\sim 30^\circ$), and depending on the lighting conditions it may be important to account for the changing view geometry across the scene. Likewise, some flight paths are such that μ_0 changes by several percent from the start of the run to the end. We can account for this effect given enough information about the path of the

[3]It is important to note that the linear interpolation on aerosol optical depth is performed so that there are *only* 9 intervals spaced equally between any two neighboring τ grid values, so there are technically only 91 different values of aerosol optical depth that can be derived.

sensor. `tafkaa_tabular` has a few different methods to improve the calculation of changing solar and view geometries, which are described in the following sections.

5.2.1 `tafkaa_geometry = 1`

Assume that the sensor is nadir viewing in the middle of the array, and that there is no roll, that the sensor follows a straight, level path, and that the cross-track is perpendicular to the sensor path. Then the view zenith angle is simply a function of the location of the pixel away from the center of the image, and the view azimuth angle changes from one value to another at the center of the array. In order to calculate both the view zenith and view azimuth angle for all the pixels, `tafkaa_tabular` needs the the pixel-to-pixel angular spacing and the heading of the sensor. `tafkaa_tabular` then divides the scene into a number of regions and uses the average zenith angle of each region to determine the atmospheric correction for that region. The necessary keywords are `tafkaa_geometry`, `tafkaa_p2p_delta_theta`, and `tafkaa_sensor_heading`. The solar geometry is calculated from the various `image_center` keywords, and is constant for the whole image.

5.2.2 `tafkaa_geometry = 110`

A more exact solution that has been implemented in the latest version of *Tafkaa*. This is a two stage solution, aiming to correct for both changing view and solar geometry. First, given a latitude, longitude, date, and time for the center of each line `tafkaa_tabular` calculates the solar zenith angle at the center of the flight line. This may be necessary for long, slow flight lines. Second, given the heading and a pointing file (that indicates the off-axis angle that each each pixels looks), one calculates a view zenith and azimuth angle for each cross-track sample. This approximation still assumes a level flight with nadir pointing at the center of the image, but it is an improvement over the method in the previous paragraph. In order to use this mode, the user must use the keywords `tafkaa_geometry = 110`, and must also specify `tafkaa_crosstrack_pointing_file` and `tafkaa_line_geometry_file`. Using this option will slow *Tafkaa* somewhat since it will need to update the solar zenith angle once per line, the relative azimuth angle twice per line, and the view zenith angle once for every pixel in the line.

The format required for the `tafkaa_crosstrack_pointing_file` is described in Appendix E. The format for the `tafkaa_line_geometry_file` is described in Appendix F.

Caveats of the current geometrical correction methods

In order to use either of the above more exact geometric options, the input image files must not be geometrically corrected since the view view zenith angle is assumed to be fixed by the cross-track pixel number. Geo-rectified images (i.e., that look like maps with north at the top) usually do not have this relation since most flight lines are not due north or south. It is generally best to complete all processing before applying the geometric correction.

It should be noted that this is still an approximate solution. Most airborne platforms have some roll and pitch. Planes that fly in the lower atmosphere may have quite a bit of each. Some paths are neither level nor straight. And sometimes, for whatever reason, some pixels are added to (or subtracted from) the edge, which will destroy *Tafkaa*'s assumption that the center pixel is nadir-viewing. Furthermore, any changes in heading tend to change the view azimuth angles, and that also requires a re-interpolation of the scattering tables. For complicated flight tracks with roll, pitch, and heading variations throughout the flight, such as circular tracks to simulate MODIS or SeaWiFS views, etc., it becomes more necessary for a view-angle-by-pixel *Tafkaa*. The big pitfall to using correct angular information for every pixel in the scene is that the absorption terms need to be recalculated and the scattering tables re-interpolated for each pixel. Some of the coding for this has already been written into *Tafkaa*, but a view-angle-by-pixel version of tafkaa is not yet available.

5.3 `tafkaa_tabular` Land-Water-Both Approach

In generating our transmittance tables, we co-incidentally also generated tables of atmospheric reflectance assuming a surface of zero reflectance. These tables are appropriate for use in atmospherically correcting ob-

servations over land, and they have the advantage of being calculated for the identical parameters (geometric and aerosol) as the atmospheric reflectance tables over the ocean. However, one still needs to determine the aerosol type and optical depth in order to use these tables. Thus, the choices are as follows:

1. Determine the aerosol type and optical depth from water/ocean areas of the scene (i.e., `tafkaa_aerosol_method > 0`). Use the same aerosol model and optical depth over the water and land, but use the ocean tables over the ocean and the land tables over the land. Ocean and land can be discriminated via a previously generated land mask. *Note: There is limited availability of this function. See the notes about the* `tafkaa_aerosol_method` *keyword.*

2. Alternately the user may input the aerosol type and optical depth for the scene (i.e., `tafkaa_aerosol_method < 0`), thus `tafkaa_tabular` does not need to determine these quantities on-the-fly. Once again, the previously generated land mask is used in order to determine whether the table appropriate for ocean or land is used.

3. When determining the aerosol type over water in pixel-by-pixel mode (i.e., `tafkaa_aerosol_method = 0`), deciding the appropriate treatment over land is problematic. Thus, in pixel-by-pixel mode, the user is allowed to enter the aerosol type, relative humidity, and optical depth (at $\lambda = 0.550\,\mu$m) to apply over land pixels.

6 Running *Tafkaa*

6.1 Source Code

Each version of *Tafkaa* has been created from a single[4] source code that can be compiled on at least three platforms, and the versions share many identical modules. Currently, it has been compiled on a PC platform with the *Microsoft® Fortran PowerStation™4* compiler, on an SGI® workstation with the *MIPS® Pro 7.3 Fortran 90* compiler, and on a Sun® workstation with the *Sun® Forte™ Fortran 95 version 6 update 2* compiler. All attempts have been made to make the source code strictly ANSI/ISO Fortran 90 [ISO/IEC 1539:1991] compliant. Since none of the deleted features are used, it is also compliant with standards of Fortran 95 [ISO/IEC 1539-1:1997] and Fortran 2003 [ISO/IEC 1539-1:2004]. Language extensions beyond Fortran 77 [ISO/IEC 1539:1978] are used, so the source code will not compile with a standard Fortran 77 compiler. The executable should run correctly when compiled with an ANSI standard Fortran 90 compiler that: 1) assumes that unformatted direct access files are "flat" binary data files with no record length information or record marks in the file[5]; 2) supports the reading and writing of 1, 2, and 4 byte integers[6]; 3) supports a common numerical model[7] in order to allow for distribution of binary lookup tables.

Version Numbering, Naming, and Availability

When *Tafkaa* is executed (and in the history section), the complete version number is printed: `tafkaa_f*.b*.s*.c*`. However, the executables for *Tafkaa* have a compilation date attached to them. Compilations can occur after new features are added, after a bug fix, after the code has been made more efficient (i.e., sped-up), or after the code has otherwise been cleaned-up for more efficient maintenance (but no effect on execution). The changes increment `f*`, `b*`, `s*`, and `c*`, respectively. The *User's Guide* need only be updated after new features are added, since features usually involve new keywords or new options to old keywords. Bugs are fixed as time allows depending on their impact. Versions with new features, bug fixes, or significant decreases in execution time will be distributed to the known user list.

Please use the contact information on the front cover of this *User's Guide* in order to report bugs or request executables of *Tafkaa*.

[4]This is *not quite* true. A single file called `name_and_version.f90` differs on each platform, usually only with the text identifying the hardware although sometimes the `version_date` may differ.

[5]This is not specified in the Fortran 90 standard, but is supported by many vendors.

[6]Only one integer representation is required in a standard conforming Fortran 90 compiler, but more are allowed; many compilers support three or four types of integers.

[7]The method of representing floating point numbers or integers using binary digits.

6.2 Invocation

To invoke or execute *Tafkaa*, it is best to feed it the input file via a "standard input redirect," which is allowed under both the command prompt window on a PC, and under most shells available on the preponderance of Unix based operating systems. Thus, if I symbolize the prompt as `prompt>`, the following is valid on both PCs and under `csh` & `tcsh`:
`prompt> path-to-tafkaa < tafkaa_input_file`.
It is always safest to use the full path to both the *Tafkaa* executable and the *Tafkaa* input file.

6.3 *Tafkaa* Inputs

The easiest way to run *Tafkaa* on any of the currently supported platforms is to use the interface for the ENVI® software application provided by W. Snyder. The graphical interface gathers the required information from the user, and prepares the image header file and the *Tafkaa* input file.

The *Tafkaa* input file is an ASCII text file made up of many keywords. The structure is

```
keyword = value
```

The values come in six basic types: integer, real, and string; integer array, real array, and string array. Array types are enclosed in curly braces, i.e., { }.

Except for `tafkaa_input_image_name`, the keywords may appear in either the input file or the image's header file. However, it is considered good form to include keywords that concern information required to read the data or concerning the actual image, to appear in the image's header file. Keywords specific to *Tafkaa* should then appear in the input file.

All lines that appear in the input file must have no more than 132 characters in the line; *lines longer than 132 characters will lead to some type of input errors, such as not finding a closing "}" to end processing of input array values.* Any values that must wrap around (such as long arrays) must have line breaks *after* a comma, after a left curly brace, or before a right curly brace.

Certain other keywords have been defined, and may occur in the header file. The lists here do *not* include all these keywords. Only the keywords *required* by *Tafkaa* are listed below. *Tafkaa* will ignore other text in the file.

If a keyword is repeated in the input file and in the header file, the value in the input file will be used.

In the following sections, items in the `teletype` font style represent what you would actually expect to see or enter in a file. Any other text, in particular, text in parentheses, is a useful comment indicating restrictions on value choices, implied units, suggested reasonable values, etc.

6.3.1 Input Image Header File

The following nine keywords should appear in the image's header file. The first seven keywords listed below are necessary and must be present in order for the binary image file to be read (for example, they must be present in the header file for the ENVI® software program to read it; *Tafkaa* merely requires their presence in either the image header file or the *Tafkaa* input file); the last two are necessary for *Tafkaa*, but are not necessary for an image analysis application to read data from the image file. The values shown are are examples only, and need to be set correctly for your particular image. An example input image header file is listed in Appendix G.1.2. The lines in the input image header file must be ≤ 132 characters.

- `samples = 614`
- `lines = 600`
- `bands = 224`
- `header offset = 0` (In bytes; *Tafkaa* requires this to be `0`.)
- `data type = 2` (2 = signed 2 byte integer, which *Tafkaa* requires)
- `interleave = bip` (`bip`, `bil`, or `bsq`.)

- `byte order = 1` (0 which is Least Significant Byte first, for PC/DEC; else 1 which is Most Significant Byte first. This refers to the input image data, *not* to the computer on which you are executing *Tafkaa*. *Tafkaa* can byte-swap on the fly.)

- `wavelength = { 0.4121, ..., 2.5090}` (Center wavelength of band, in μm, one element for each band)

- In order to calculate the instrumental response function, one of the following two keywords needs to be provided. Either

  - `fwhm = { 0.009691, ..., 0.010030}` (in μm, assuming Gaussian instrumental response function, one element for each band)

  or

  - `tafkaa_instrumental_response_file = complete_path_to_file` (Filename with the complete path to the instrumental response file. The file should be as described in Appendix D.)

The following items should also appear in the image's header file, since it is the logical location for them. These keywords (all begin with `tafkaa_*`, `image_*`, or `sensor_*`) are non-standard for any image display application. In some instances, the keywords may disappear if the header file is edited from within the image display application; in other instances, the image display application will merely ignore any keywords it does not understand. Because of this, an argument can be made for the following keywords to be put in the input file instead of the image header file. Current practice is to include the following image-specific keywords in the image's header file, and remind the user to be cautious when editing the header file from within any image processing application.

- `image_scale_factor= { 100., ..., 50.}` (Either 1 element if all N_B elements have the same scale factor f_i, or N_B elements if they don't. We divide these numbers into the integers in the image file in order to obtain the physical, measured radiance in units of W m^{-2} sr^{-1} μm^{-1}: $L_{\rm obs} = N_i/f_i$.)

- `image_center_date = {yyyy, mm, dd}` (GMT date, assuming standard civil Gregorian calendar, $1 \leq$ `mm` ≤ 12, and $1 \leq$ `dd` ≤ 31.)

- `image_center_time = {hh, mm, ss.ss}` (GMT time, $0 \leq$ `hh` < 24; $0 \leq$ `mm` < 60; $0 \leq$ `ss.sss` < 60)

- `image_center_long = {ddd, mm, ss.sss}` (non-negative, degrees, minutes, seconds; $0. \leq$ `ddd` ≤ 180.)

- `image_center_long_hem = W` (Either `W` or `E` of the Prime Meridian)

- `image_center_lat = {dd, mm, ss.sss}` (non-negative, degrees, minutes, seconds; $0. \leq$ `dd` ≤ 90.)

- `image_center_lat_hem = N` (Either `N` or `S` of the Equator)

- `image_center_zenith_ang = {dd, mm, ss.sss}` (view zenith angle at the center of the image, non-negative, degrees, minutes, seconds; { 0., 0., 0.} implies nadir-viewing; $0. \leq$ `dd` ≤ 90, however due to the plane-parallel atmosphere of the radiative transfer calculations, the practical upper limit is 72°.)

- `image_center_azimuth_ang = {dd, mm, ss.sss}` (view azimuth angle at the center of the image, non-negative, degrees, minutes, seconds, relative to north, clockwise; $0. \leq$ `dd` $< 360.$)

- `sensor_altitude = 20.0` (kilometers above mean sea level)

- `tafkaa_ground_elevation = 0.00` (Average elevation of viewed surface in kilometers above sea-level.)

The values for latitude, longitude, view zenith, and view azimuth may also be written as {`dd.ddd, 0.0, 0.0`} if you have fractional degrees available, or even {`dd, mm.mmm, 0.000`} if you have fractional minutes available. The values for arc-minutes and arc-seconds have valid ranges of $0 \leq$ `mm` < 60 and $0 \leq$ `ss.sss` < 60.

6.3.2 *Tafkaa* Input File

The following items are specific to *Tafkaa*, and thus should appear in the *Tafkaa* input file. An example input file listed in Appendix G.1.1.

The defining directories, files, masks, and output format:

- `tafkaa_input_image_name = /u1/mmontes/coral.bip`
- `tafkaa_data_directory = /u1/mmontes/data/` (The directory that contains the high resolution exo-atmospheric solar irradiance spectrum in units of W m^{-2} sr^{-1} μm^{-1}. The data is in a binary file named `sun_binary` that consists of reals and is expected to be in the native byte order. This directory also contains all the files containing the line absorption coefficients and the various tables needed by the tabular version.)
- `tafkaa_output_root_name = /u1/mmontes/testing` (The name, including directory path, to which is appended various suffices in order to form the various output image and header file names.)
- `tafkaa_use_which_masks = { land }` (Select either `none` or one or more of `land`, `cirrus`, `low cloud`, `bad pixel` in any order. This tells *Tafkaa* which masks to utilize. Any pixels that are flagged as cirrus, low cloud, or bad pixels are blanked on output. Land is treated as described in the discussion of the `tafkaa_aerosol_method` keyword on page 13.)
- `mask_image_name = ./coral_test_mask.img` (The name of the mask image that has been previously produced; not necessary if `tafkaa_use_which_masks = { none }`. Currently, *Tafkaa* is set up to read masks that are formatted a certain way. The data must be one-byte integers, with a value of 100 (0) indicating the presence (absence) of the named mask at each pixel. The case sensitive `band names` for the various bands listed above must be `Land Mask`, `Cirrus Mask`, `Low Altitude Cloud Mask`, and `Bad Pixel Mask`, respectively.)
- `tafkaa_output_type = refl` (One of `refl, rrs, lgn, lg`, or `aprefl`, which imply reflectance (ρ_g), remote-sensing reflectance (R_{rs}), normalized ground-leaving radiance ($[L_g]_N$), ground-leaving radiance (L_g), and apparent at-sensor reflectance (ρ^*_{obs}), respectively. *The* `lg` *option is not currently supported.* The apparent at-sensor reflectance (see Equation 5) does not perform any atmospheric correction at all, and the various atmospheric and aerosol keywords are not necessary in the input file. The apparent at-sensor reflectance is a useful diagnostic tool.)
- `tafkaa_output_scale_factor = 10000.` [In order to get to the relevant units, divide the integers in the output image by this number. The *best* output scale factor to use depends on the output quantity, and aesthetics. Let me define `tafkaa_output_scale_factor` $=\chi$, and the maximum value of the chosen quantity (ρ, R_{rs}, $[L_g]_N$, or L_g) will be represented as M. Since we are representing numbers as signed 2-byte integers, the maximum number we can have is $2^{15}-1 = 32767$. Thus, usually we want $\chi \leq 32767/M$. Since $\rho \leq 1 = M$, in this case, $\chi_\rho <= 32767$. Aesthetics plays a role since many people prefer powers of 10, so a typical value for this case would be $\chi_\rho = 10000$. In the case of $R_{rs} = \rho/\pi$, $M = 1/\pi$, thus $\chi_{R_{rs}} \leq 32767\pi = 102940.57$. This would allow a nice aesthetic value of $\chi_{R_{rs}} = 100000$. $[L_g]_N = E_0 R_{rs} \leq 2000/\pi$ thus in this case $M \sim 640$. E_0 is wavelength dependent, and it peaks at about $0.48\,\mu$m; thus the resolution will be better in some bands than in others. This implies that $\chi_{[L_g]_N} \leq 51$. Finally, $L_g = [L_g]_N \mu_0 t_d$. Clearly, $\mu_0 \leq 1$ and $t_d \leq 1$. Away from the absorption bands, $t_d \sim 1$. μ_0 is known, and is usually not too close to unity. Thus, $L_w \leq 640/\mu_0$, so $\chi_{L_g} \leq 51/\mu_0$.]
- `tafkaa_line_range = { start_line, stop_line}` (Optional; integer array. The range of lines to process, allowing *Tafkaa* to process pieces of a scene; by default, all lines are processed if this keyword is absent. `y start` is respected: the values are assumed to be in the system where the first line is identified by the `y start` keyword in the header. Thus, the values in this keyword must be such that `y start` $\leq$ `start_line` $\leq$ `stop_line` $\leq$ `y start + lines - 1`. The output files will have `y start = start_line` and `lines = stop_line - start_line + 1`.)
- `tafkaa_sample_range = { start_sample, stop_sample}` (Optional; integer array. The range of samples to process, allowing *Tafkaa* to process pieces of a scene; by default, all samples are processed

if this keyword is absent. `x start` is respected: the values are assumed to be in the system where the first sample is identified with the `x start` keyword in the header. Thus, the values in this keyword must be such that `x start` $\leq$ `start_sample` $\leq$ `stop_sample` $\leq$ `x start + samples - 1`. The output files will have `x start = start_sample` and `samples = stop_sample - start_sample + 1`.)

The following keywords modify how the solar and view geometry are computed; see the discussion in §5.2:

- `tafkaa_geometry = 1`

  1. Set to `0` for a single geometry for the whole scene; in this case, `image_center_zenith_ang` and `image_center_azimuth_ang` are used.
  2. Set to `1` in order to use multiple view angles across a scene, and see below for the required keywords for this case; see §5.2.1.
  3. Set to `10` in order to correct for view-geometry across a scene, assigning each cross-track pixel a particular view angle. This requires the use of `tafkaa_cross_track_pointing_file` and `tafkaa_line_geometry_file`.
  4. Set to `100` in order to correct for changing solar geometry in the along-track (line number) direction. This requires the use of `tafkaa_line_geometry_file`.
  5. Set to `110` in order to correct for the changing solar geometry in the along-track direction, and the view geometry in the cross-track direction. This requires the use of `tafkaa_cross_track_pointing_file` and `tafkaa_line_geometry_file`; see §5.2.2.
  6. If this keyword is not present, `tafkaa_geometry = 0` is set.

- `tafkaa_p2p_delta_theta = 0.8745e-3`
  (Used if `tafkaa_geometry = 1`; the cross-track pixel-to-pixel spacing, in radians.)

- `tafkaa_sensor_heading = 260.0` (Used if `tafkaa_geometry = 1`; the sensor heading in degrees from north.)

- `tafkaa_cross_track_pointing_file = ./aviris_pointing.txt` (The file name of the cross-track pointing file, used when `tafkaa_geometry` is `10` or `110`. The format of this file is described in Appendix E.)

- `tafkaa_line_geometry_file = ./f010731_r04_line_geometry.txt` (The file name of the line-geometry file, used when `tafkaa_geometry` is `10`, `100`, or `110`. The format of this file is described in Appendix F.)

Parameters that describe the absorptive properties of the atmosphere:

- `tafkaa_atmo_model = tropical`
  (Choices are `tropical`, `mid latitude summer`, `mid latitude winter`, `subarctic summer`, `subarctic winter`, and `US Standard 1962`.)

- `tafkaa_atmo_gasses = { H2O, CO2, O3, N2O, CO, CH4, O2}`
  (One or more of `H2O`, `CO2`, `O3`, `N2O`, `CO`, `CH4`, `O2`, or `NO2`, in any order.)

- `tafkaa_atmo_ozone = 0.34` (Measured in units of atm-cm, and required if `O3` is selected in `tafkaa_atmo_gasses`. An online resource for ozone values is: `http://toms.gsfc.nasa.gov/ozone/ozone.html` or `http://toms.gsfc.nasa.gov/teacher/ozone_overhead.html`.)

- `tafkaa_atmo_no2_scale = 1.0` (Multiplicative factor: multiplies $5{\times}10^{15}$ molecules NO_2; required if `NO2` is selected in `tafkaa_atmo_gasses`.)

- `tafkaa_atmo_clmwvap = 1.1` (Vertical column of water vapor, in centimeters. **Including a positive value for this keyword implies the next 5 keywords listed below are unnecessary, and will be ignored if included in the input file.** You *must* include `H2O` in the `tafkaa_atmo_gasses` list in order for this value to be used.) **Do not use this keyword if `tafkaa_h2o_enter_inputs = 1`.**

- `tafkaa_h2o_enter_inputs = 1` [`1` (`0`) implies the user will (won't) enter the parameters for the water vapor bands, described in the next four parameters.] **Note 1: Only necessary when determining column water vapor pixel-by-pixel. Note 2: Do not use `tafkaa_atmo_clmwvap` keyword when `tafkaa_h2o_enter_inputs = 1`. Note 3: Using the value `0` automatically chooses the below 4 parameters to be those associated with the 0.94 μm absorption band and the 1.14 μm absorption band.**

- `tafkaa_h2o_wl_set1 = {0.8650, 1.030, 0.945}` (Wavelengths in μm of the two adjacent windows and the center of the H_2O absorption band. See Gao et al. (1993) for appropriate selection of this parameter for snow, vegetation, or bare surfaces. The wavelengths listed in for `set1` may be the same as for `set2`, and/or may use other water vapor features than listed here.)

- `tafkaa_h2o_nb_set1 = {3, 3, 5}` (Number of bands to use centered at each of the above wavelengths. The number of bands chosen depends on the particular water vapor that is chosen, and on both the spectral resolution and spectral spacing of the instrument. See Gao et al. (1993) for appropriate selection of this parameter for snow, vegetation, or bare surfaces.)

- `tafkaa_h2o_wl_set2 = {1.0650, 1.240, 1.140}` (Wavelengths in μm of the two adjacent windows and the center of the H_2O absorption band. See Gao et al. (1993) for appropriate selection of this parameter for snow, vegetation, or bare surfaces. The wavelengths listed in for `set2` may be the same as for `set1`, and/or may use other water vapor features than listed here.)

- `tafkaa_h2o_nb_set2 = {3, 3, 5}` (Number of bands to use centered at each of the above wavelengths. The number of bands chosen depends on the particular water vapor that is chosen, and on both the spectral resolution and spectral spacing of the instrument. See Gao et al. (1993) for appropriate selection of this parameter for snow, vegetation, or bare surfaces.)

- `tafkaa_use_prev_atmo_trans = 1` [`1` (`0`) implies you do (don't) want to use the gas absorption results from a previous calculation; see §6.4]

- `tafkaa_prev_atmo_trans_file = ./coral_test_tafkaa.vapbin` (If `tafkaa_use_prev_atmo_trans = 1`, then you must set this value to the filename that contains the results from a previous calculation; see §6.4)

- `tafkaa_atmo_gas_scale_factors = {1.0, 1.0, 1.0, 1.0, 1.0}` (Optional. Multiplicatively scale the amount of {CO_2, N_2O, CO, CH_4, O_2 }, respectively. If this keyword is not present, all values are assumed to be unity. It is best to not fiddle with this keyword.)

Parameters that describe the aerosol model and/or how the aerosol model(s) are chosen ***for the tabular version only***:

- `tafkaa_aerosol_method = 0` [`0` implies attempt a pixel-by-pixel solution; any negative value implies the user will enter an aerosol model described by the next four parameters; a value > 2 value implies use that number of pixels to determine the aerosol type, then apply it over the whole scene (*this last option is not yet available*).]

  If `tafkaa_aerosol_method = 0`, then the following three parameters describe the model that will be applied over the land pixels (as described by the land mask) if they are supplied; if the following parameters are not supplied when `tafkaa_aerosol_method = 0`, then no models will be applied over the land and the land will be blacked out. If `tafkaa_aerosol_method < 0`, then this model will be applied over the whole scene. **(The choices are different for the 6S version. The choices for the 6S version are described following this description appropriate for the tabular version.)** The choices for the tabular version are:

  - `tafkaa_aerosol_rh = 80%` (Choose one of `50%`, `70%`, `80%`, `90%`, or `98%`.)
  - `tafkaa_aerosol_model = coastal-a` (Choose one of `maritime`, `coastal`, `coastal-a`, `tropospheric`, or `urban`.)
  - `tafkaa_aerosol_tau550 = 0.1` (Choose a value in the range $0 \leq \tau_{550\,\mathrm{nm}} \leq 2.0$.)

If `tafkaa_aerosol_method = 1` or `tafkaa_aerosol_method = 2` then *Tafkaa* will determine the aerosol type and optical depth using a rectangular area previously selected by the user. `tafkaa_aerosol_method = 1` determines $\rho^*_{\rm obs}/T_{\rm g}$ for each pixel, averages over the rectangular region, and uses that quantity to determine the aerosol type and optical depth. `tafkaa_aerosol_method = 2` determine $\rho^*_{\rm obs}$ for each pixel, uses that to determine an average water-vapor value, then determines the aerosol model and type from $\overline{\rho^*_{\rm obs}}/T_{\rm g}^{\rm rectangle}$. The coordinates of the rectangular are are given in the keyword

- `tafkaa_aerosol_rectangular_region = {llx, lly, urx, ury}` (`llx` is the lower-left x-coordinate of the region; `lly` is the lower-left y-coordinate of the region; `urx` is the upper-right x-coordinate of the region; `ury` is the upper-right y-coordinate of the region. The coordinate should be measured assuming the `x start` and `y start` values of the header. If either `x start` or `y start` is not present, the value is assumed to be unity. If `tafkaa_geometry = 1`, then it must be the case that `sample_range(1)` $\leq$ `llx` $\leq$ `urx` $\leq$ `sample_range(2).`)

When `tafkaa_aerosol_method`≥ 0, *Tafkaa* attempts to determine an aerosol model. With the addition of the urban aerosol model to our tables, there are some situations where *Tafkaa* chooses the urban aerosol in places where we know it can't be present – apparently there is a slight degeneracy between some of the higher optical depth urban aerosol models and some of the lower optical depth non-urban models. In order to deal with this, we've instituted a few keywords to exclude choosing certain aerosols or optical depths. The following lists to exclude are "or"ed together.

- `tafkaa_exclude_aerosol_models = { urban }` (A string array accepting the names of aerosol models you wish to exclude. If this keyword is not listed, then no aerosol models are excluded.)
- `tafkaa_exclude_aerosol_rh = { 80%, 50% }` (A string array accepting a list of relatvive humidities you wish to exclude. If this keyword is not listed, then no aerosol relative humidities are excluded.)

On output, the determined aerosol type and optical depth will be reported in the history section of the output header files.

- `tafkaa_wind_speed = 2` (*Not* used for interpolation, effectively chooses which table to use: the 2 m/s, 6m/s, or 10 m/s table. The units are m/s.)
- `tafkaa_aerosol_weights = {0, 0, 0, 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.}` (Used only when `tafkaa_aerosol_method` $\geq$ `0`; a set of 14 reals, associated with the wavelengths 0.39, 0.41, 0.44, 0.47, 0.51, 0.55, 0.61, 0.67, 0.75, 0.865, 1.04, 1.24, 1.64, and 2.25 μm, respectively. Chooses which parts of the spectra to use when determining the aerosol model, relative humidity, and optical depth.)
- `tafkaa_interp_sensor_altitude = true` (Possible values are `true`; anything except `true` implies `false`. Use this when `sensor_altitude` is anything inside the atmosphere. A value of `true` implies that we'll use linear interpolation in pressure to determine the sensor values appropriate for the sensor height. A value of `false` implies that we'll use the nearest altitude. This may be used when you fly at one of the altitudes the tables have been generated at, and you don't want to interpolate. Anytime you **want** to interpolate, set this keyword to **true**. The current altitude values of the aerosol tables are: 84.0, 22.0, 16.0, 10.5, 8.0, 5.5, 4.0, 2.59, 2.50, and 0. kilometers. Any value greater than or equal to 84.0 km is treated as above the atmosphere and uses the same tables no matter what: no interpolation is performed.)

The following **required** parameters describe the aerosol model ***for the 6S version only***:

- `tafkaa_aerosol_model = maritime` (Choose one of `none`, `continental`, `maritime`, or `urban`.)
- `tafkaa_aerosol_visibility = 50.0` (The visibility in kilometers; put 0 here if you would rather use `tafkaa_aerosol_tau550`.)
- `tafkaa_aerosol_tau550 = 0.1` (Used only when `tafkaa_aerosol_visibility = 0.0`; a reasonable range $0 \leq \tau_{550\,\rm nm} \leq 2.0$. Note: Check the range!)

6.4 Output Files

An output image file, along with an output data products file, are always created; their header files will contain a "history" section that indicates that version and the settings that were used to to generate them. Additionally, there are two additional ASCII files and possibly a binary file that may be output. All the output files will be described below.

The principal output image file and its header have the following names:

- `tafkaa_output_root_name`_tafkaa_refl.hdr, and
- `tafkaa_output_root_name`_tafkaa_refl.img

if reflectance was chosen as the output type. When remote-sensing reflectance is the output type, 'refl' will be replaced with 'R_rs'. When normalized ground-leaving radiance is selected, 'refl' will be replaced with 'NLsf'. Finally, when ground-leaving radiance is selected, 'refl' will be replaced with 'Lsfc'. The file will have the same storage order and number of bands as the input observed radiance file had, and it will always be in the machine's native byte-order in two-byte integers. The values of `samples`, `lines`, `x start`, and `y start` in the output file depend on the values of the input parameters `tafkaa_line_range` and `tafkaa_sample_range`. Frequently, a bad bands list keyword (`bbl`) will be added. This is used to mask out the regions near the 1.4 and 1.88 μm water vapor transition. These transitions are nearly fully absorbed. Dividing by the T_g in this region implies division by a small number; slight errors in claculating the transmission cause difficulty in correcting these regions of the spectrum. This information is provided in the header file.

The products file and its header have the following names:

- `tafkaa_output_root_name`_tafkaa_prod.hdr, and
- `tafkaa_output_root_name`_tafkaa_prod.img .

This file is always made up of two-byte integers in the machine's native byte-order in BSQ sort-order, with the same number of samples and lines as the principle output image (above). The first plane is always the derived water-vapor plane. When the *tabular* version is used and `tafkaa_aerosol_method = 0`, there will be additional planes that contain $\tau_{\text{aerosol } 550\,\text{nm}}$, relative humidity, and the aerosol model. There may also be a plane with some sort of measure of the quality of the fit, especially in the early development of *Tafkaa*. All this information along with the appropriate scale factors is provided in the header file.

The two ASCII files that are output are:

- `tafkaa_output_root_name`_tafkaa_solar_irr.asc, and
- `tafkaa_output_root_name`_tafkaa_vaplib.asc .

The three columns in the first file are λ (in μm), the top-of-the-atmosphere E_0, and $\mu_0 E_0$ (the latter two in W m^{-2} μm^{-1}). The second file contains an ASCII version of the T_g lookup tables, convolved to the sensor's IRF.

If `tafkaa_use_prev_atmo_trans = 0` or if `tafkaa_use_prev_atmo_trans = 1` and `tafkaa_prev_atmo_trans_file` is not appropriate for the atmospheric parameters and geometry that the user indicates, a binary file called

- `tafkaa_output_root_name`_tafkaa.vapbin

will be written for the atmospheric parameters specified by the user. This file is valuable if the user is experimenting with many different aerosol types, but the geometry (date, time, view angles, and solar angles) and atmospheric parameters are constant.

7 Frequently Provided Answers: Useful Notes & Items to Consider Before Using *Tafkaa*

This section contains answers that are frequently provided to new users of *Tafkaa* and is based on our interactions with many researchers using *Tafkaa* on many different data sets and with many different computers. Additionally, this section has been used to provide otherwise interesting information that does not easily fit into other parts of the document, or to emphasize points made elsewhere in this *User's Guide*.

1. Based on the input files (solar, absorptive, and lookup tables), the applicable range of *Tafkaa* is $0.370\,\mu\text{m} < \lambda < 2.55\,\mu\text{m}$.

2. *Tafkaa* should be limited to images that have $\theta_0 \leq 72^\circ$. θ_0 at the center of the image is written to the history section of the output header files. There are many utilities available on the internet that allow quick calculation of θ_0.

3. There are also limits on θ. However, these are more dependent on the sensor altitude. The primary effects arise from the curvature of the earth (i.e., not plane parallel) and refractive effects of the atmosphere. A useful practical limit is $\theta \leq 72^\circ$.

4. The discretization of the independent variables in the lookup-table grids should allow for retrievals of reflectance to within $\lesssim 0.001$.

5. Input pixels that have
$$\text{all}_{i=1}^{N_B}(N_i \leq 0)$$
are not processed, and will be output as blank pixels with values at all bands set to 0.

6. It is best if the image data is formatted so that the first line of data is earlier in time than the second, i.e., so that as time increases, the line number increases. The cross-track samples should be ordered so that the displayed image looks like what you would see if you were looking through a window in the bottom of the plane or sensor. Most data is delivered to the user in such a format already. Data that is not in this format *might* have strange results when using any geometry option other than `tafkaa_geometry = 0`.

7. *Tafkaa* can be used on data sets that have been geometrically corrected. You should not use `tafkaa_tabular`'s ability to correct for solar and view geometry in this case, as the assumptions will no longer make any sense. The previous two items in this list are relevant when running *Tafkaa* on geometrically corrected data.

8. Any `default bands` keyword in the input file will be repeated in the output file. If it is not present in the output file, it will be set to bands near $0.670\,\mu\text{m}$, $0.55\,\mu\text{m}$, and $0.44\,\mu\text{m}$.

9. There are some diagnostics available. *Tafkaa* already prints out ASCII text files of the solar irradiance at the top of the atmosphere, (in `*tafkaa_vaplib.asc`) as well as the gaseous transmission information (in `*tafkaa_vaplib.asc`), both convolved to the sensor's IRF. If a fixed aerosol is chosen (`tafkaa_aerosol_method = -1`), a single geometry is used (`tafkaa_geometry = 0`), and a fixed amount of water vapor is chosen, then *Tafkaa* will print a scattering diagnostic file (`*_tafkaa_diag_refl.asc`).

10. The ordering of the data affects the speed of execution. All else being equal, BIP data is processed the fastest, and BSQ data is processed the slowest. Data ordering is a compromise of all uses of the data, however. *Tafkaa* may execute fast enough that the sort order (the `interleave`) of the data does not matter for the particular data set that you use. As the data set becomes larger, the absolute time difference of *Tafkaa* runs on data of different interleaves will increase.

11. The byte order of the data affects the speed of execution. All else being equal, data in the byte order that is the same as the native byte order of the computer you are using will execute faster than if *Tafkaa* needs to byte-swap the data as it is read. *Tafkaa* may execute fast enough that the time difference does not matter for your particular data set.

12. ASCII text files (such as header files and *Tafkaa* input files) need to be carefully transferred between Unix and non-Unix platforms, as the end of line character differs. Ensure that FTP transfers of text files is performed using ASCII mode. Take the time to open the text files and verify that there are no strange characters at the end of the each line.

13. Ensure that the last line of the header and tafkaa input files has an end-of-line appropiate to your operating system. Or, better yet, just add a blank line at the end of an ASCII text file that *Tafkaa* will be reading.

14. *Tafkaa* is case sensitive when reading keywords and values. Ensure you are using the correct case as shown in the sections above. There is a difference between the number one (`1`) and the lower-case letter "L" (`l`). There is also a difference between the number zero (`0`), and the upper- and lower-case "O" (`O` and `o`, respectively). If *Tafkaa* does not find a keyword that you know is in the header or input file, check each keyword for correct spelling.

15. With operating systems, hardware, or compilers that use signed 32-bit addressing there is no way to address files that are larger than $2^{31} - 1 = 2147483647$ bytes, $\approx$ 2.1GB (where kB = 10^3B, not 1024B). This issue is most noticeable for some PCs. Many recent computers do not have this problem since they use signed 64-bit integers, which implies a maximum file size of $2^{63} - 1$ bytes, $\approx$ 9.2EB. There are other addressing schemes in use that allow access to files larger than $\approx$ 2.1GB, but still smaller than $\approx$ 9.2EB. Users that apply *Tafkaa* to files larger than the operating system can handle will get unpredictable results that depend on the hardware, operating system, and probably the version of the compiler that was used to make the executable. Past symptoms have been that *Tafkaa* just seems to stop processing (sometimes with exiting, sometimes without) at a location (calculated from line number, sample, and bands) $\approx$ 2.1 GB into the input file.

16. *Tafkaa* cannot fix bad data. Using fully absorbed transitions to determine the amount of water vapor is dangerous, in the sense that you'll get bizarre results. So is using transitions that are not fully calibrated and/or transitions where *negative* values of radiance occur (noisy data, for example). In all these cases the retrieved value of water vapor will be wrong for the particular bad pixels, and so all the water transitions in the spectra of those pixels will be corrected incorrectly– i.e, they'll be mis-corrected. We are considering the need to write out another plane of information to note this problem, but it is currently not implemented.

17. The best transitions to use for determining the amount of water vapor are the 0.914 μm and 1.14 μm absorption bands. If these are not available in your hyperspectral imagery, it may be necessary to use the 0.820 μm feature. Lack of good (or any) data may require the duplication of the parameters for the `set1` and `set2` keywords.

18. The retrieval of the amount of water vapor over dark surfaces such as the ocean is biased low, since few of the photons are reflected from the surface itself. The amount used should best remove the water vapor features in the spectrum, but will not accurately represent the amount of water vapor present

19. *Tafkaa* may appear to overcorrect in the blue. Frequently the issue is related to the calibration of the instrument. One test is to run *Tafkaa* with no aerosols: set it to a fixed aerosol, choose a model and relative humidity (it does not matter which one), and set the aerosol optical depth to 0. If the output is still negative in the blue, it is a good indication that there are calibration problems with the data.

20. *Tafkaa* does not correct well over swells. The geometry may be too different from a flat surface over the faces and slopes of some swells. This is affected by the orientation and slopes of the swells as well as the solar and view geometry. In pixel-to-pixel mode, areas that appear brighter are assigned a higher optical depth– clearly this is not correct. This is compicated problem, and possible solutions are under discussion.

21. If the true physical aerosol optical properties are substantially different than those in the models, the retrievals won't be as accurate.

22. And finally, *Tafkaa* assumes that we're observing the Earth (for atmospheric properties, latitude, longitude grid) at the current epoch (i.e., with the sun at its current luminosity). Don't try to use it for data from other planets.

Acknowledgments

The development of *Tafkaa* and of this *User's Guide* was supported by funding from the Office of Naval Research.

Tafkaa was generated using many pieces of B.-C. Gao's *ATREM*, including most of the geometry, solar-spectral, and water vapor subroutines. At a minimum, any of the subroutines from *ATREM* used in *Tafkaa* have been rewritten by converting them to Fortran 90, and all incorporate modifications that allow them to execute much more quickly, as well as other algorithmic modifications. Many subroutines in *Tafkaa* have been written entirely from scratch.

Many thanks to Bill Snyder, Dave Kohler, and Rob Steward for their helpful suggestion for new features, and for finding and reporting bugs.

This document was typeset using LaTeX 2_ε.

Legal

Bibliography

Z. Ahmad and R. S. Fraser. An iterative radiative transfer code for ocean-atmosphere systems. *J. Atmos. Sci.*, 39:656–665, 1982.

Curtiss O. Davis, Jeffrey Bowles, Robert A. Leathers, Dan Korwan, T. Valerie Downes, William A. Snyder, W. Joe Rhea, Wei Chen, John Fisher, W. Paul Bissett, and Robert Alan Reisse. Ocean PHILLS hyperspectral imager: design, characterization, and calibration. *Optics Express*, 10(4):211–221, February 2002.

R. S. Fraser, S. Matoo, E.-N. Yeh, and C. R. McClain. Algorithm for atmospheric and glint corrections of satellite measurements. *J. Geophys. Res.*, 102:17107–17118, 1997.

B.-C. Gao and C. O. Davis. Development of an operational algorithm for removing atmospheric effects from HYDICE and HSI data. In Michael R. Descour and Jonathan M. Mooney, editors, *Imaging Spectometry II*, volume 2819 of *Proc. SPIE*, pages 45–55, 1996.

B.-C. Gao and C. O. Davis. Development of a line-by-line based atmosphere removal algorithm for airborne and spaceborne imaging spectrometers. In M. R. Descour and S. S. Shen, editors, *Imaging Spectrometry III*, volume 3118 of *Proc. SPIE*, pages 132–141, 1997.

B.-C. Gao, K. Heidebrecht, and A. F. H. Goetz. *Atmospheric Removal Program (ATREM) Users Guide, Version 3.0.* Center for the Study of Earth from Space, University of Colorado, Boulder, Colorado, 1997.

B.-C. Gao, K. H. Heidebrecht, and A. F. H. Goetz. Derivation of scaled surface reflectances from AVIRIS data. *Remote Sens. Env.*, 44:165–178, 1993.

B.-C. Gao, M. J. Montes, Z. Ahmad, and C. O. Davis. Atmospheric correction algorithm for hyperspectral remote sensing of ocean color from space. *Appl. Opt.*, 39(6):887–896, 2000.

ISO/IEC 1539:1978. Fortran 77.

ISO/IEC 1539:1991. Fortran 90.

ISO/IEC 1539-1:1997. Information technology - Programming languages - Fortran - Part 1: Base Language. Fortran 95.

ISO/IEC 1539-1:2004. Information technology - Programming languages - Fortran - Part 1: Base Language. Fortran 2003.

F. X. Kneizys, E. P. Shettle, L. W. Abreu, J. H. Chetwynd, G. P. Anderson, W. O. Gallery, J. E. A. Selby, and S. A. Clough. Users Guide to LOWTRAN7. Technical Report AFGL-TR-8-0177, Air Force Geophys. Lab., Bedford, MA, 1988.

L. Kou, D. Labrie, and P. Chylek. Refractive indices of water and ice in the 0.65–2.5 μm spectral range. *Appl. Opt.*, 32:3531–3540, 1993.

C. D. Mobley. Estimation of the remote-sensing reflectance from above surface measurements. *Appl. Opt.*, 38:7442, 1999.

M. J. Montes, B.-C. Gao, and C. O. Davis. A new algorithm for atmospheric correction of hyperspectral remote sensing data. In William E. Roper, editor, *Geo-Spatial Image and Data Exploitation II*, volume 4383 of *Proc. SPIE*, pages 23–30, Orlando, FL, 2001.

M. J. Montes, B.-C. Gao, and C. O. Davis. Tafkaa atmospheric correction of hyperspectral data. In Sylvia S. Shen and Paul E. Lewis, editors, *Imaging Spectrometry IX*, volume 5159 of *Proc. SPIE*, pages 188–197, San Diego, CA, 2003.

W. Ridgway, personal communication, 1996. Code 913, NASA Goddard Space Flight Center, Greenbelt, MD, 20771.

E. P. Shettle and R. W. Fenn. Models for aerosols of the lower atmosphere and the effects of humidity variations on their optical properties. Technical Report AFGL-TR 790214, U.S. Air Force Geophysics Laboratory, Hanscom Air Force Base, Mass., 1979.

S. Solomon, R. W. Portmann, R. W. Sanders, J. S. Daniel, W. Madsen, B. Bartram, and E. G. Dutton. On the role of nitrogen dioxide in the absorption of solar radiation. *J. Geophys. Res.*, 104(D10):12047–12058, May 1999.

G. Vane, editor. *Airborne visible/infrared imaging spectrometer (AVIRIS)*, volume 87-38 of *JPL Publ.*, Jet Propul. Lab, Pasadena, CA, 1987. Jet Propul. Lab.

G. Vane, R. O. Green, T. G. Chrien, H. T. Enmark, E. G. Hansen, and W. M. Porter. The Airborne Visible Infrared Imaging Spectrometer. *Remote Sens. Environ.*, 44:127–143, 1993.

E. Vermote, D. Tanré, J. L. Deuzé, M. Herman, and J.-J. Morcrette. *Second simulation of the satellite signal in the solar spectrum (6S), 6S User's Guide Version 1.* NASA-GSFC, Greenbelt, MD, 1994.

Eric F. Vermote, Didier Tanré, Jean Luc Deuzé, Maurice Herman, and Jean-Jaques Morcrette. Second Simulation of the Satellite Signal in the Solar Spectrum, 6s: An Overview. *IEEE Trans. Geosci. Remote Sens.*, 35(3):675–686, May 1997.

WMO (1986). A preliminary cloudless standard atmosphere for radiation computation. Technical Report WCP 112, WMO/TD No. 24, World Meteorological Organization, 1986.

A Acronyms

6S Second Simulation of the Satellite Signal in the Solar Spectrum. A radiative transfer code, see Vermote et al. [1997, 1994].

ANSI American National Standards Institute. See `http://www.ansi.org`.

ATREM ATmospheric REMoval. An atmospheric correction algorithm; version 3.0 is described in Gao et al. [1993, 1997], Gao and Davis [1996], and version 4.0 in Gao and Davis [1997].

AVIRIS Airborne Visual Infrared Imaging Spectrometer. A hyperspectral instrument described in Vane [1987], Vane et al. [1993]. It is modified and improved every year, and documentation is best found at `http://aviris.jpl.nasa.gov`.

BIL Band Interleaved by Line. Spectral data is stored a complete line at a time: first one band for the whole line, then the second band for the whole line, etc. In Fortran, an array representing the image has the dimensions (N_S, N_B, N_L). This storage order represents a compromise between BIP and BSQ formats.

BIP Band Interleaved by Pixel. Data storage of a complete spectrum for each pixel. In Fortran, an array representing the image has the dimensions (N_B, N_S, N_L). This storage order allows for easiest spectral access to the data.

BSQ Band Sequential. Each band is stored as a whole image. In Fortran, an array representing the image has the dimensions (N_S, N_L, N_B). This storage order allows for easiest spatial access (image display) of the data.

ISO International Standards Organization. See `http://www.iso.org`.

MODIS Moderate Resolution Imaging Spectroradiometer. A multispectral instrument on two orbiting spacecraft (*Terra* and *Aqua*), each in a sun-synchronous orbit, see `http://modis.gsfc.nasa.gov/`.

NIR Near Infrared. A reference to the 0.67 – 1.0 μm spectral range.

PHILLS Portable Hyperspectral Imager for Low-Light Spectroscopy. A series of push-broom hyperspectral imagers made at the Naval Research Laboratory. See, for example, Davis et al. [2002] and `http://rsd-www.nrl.navy.mil/7230/phills.htm`.

SeaWiFS Sea-viewing Wide Field-of-view Sensor. A multi-spectral instrument on the *SeaStar* spacecraft in a sun-synchronous orbit, see `http://seawifs.gsfc.nasa.gov/SEAWIFS.html`.

SWIR Short-wave Infrared. A reference to the 1.0 – 2.5 μm spectral range.

VNIR Visible Near Infrared. A reference to the visible (0.4 – 0.67 μm) and near infrared portions of the spectrum

B Glossary

θ, ϕ	=	polar and azimuth angles of the line of sight at the sensor
θ_0, ϕ_0	=	polar and azimuth angles of the direct sunlight
λ	=	wavelength
W	=	surface wind speed
C	=	the set of parameters describing all in-water processes
τ_{a}	=	aerosol optical depth and model
L_{t}	=	total observed radiance at satellite
L_0	=	radiance above the atmosphere if the radiance just above the sea surface was 0
L_{sfc}	=	radiance of direct and diffuse light specularly reflected off the sea surface
L_{g}	=	radiance reflected of from the target (assumed to be Lambertian)
L_{w}	=	radiance of water leaving light scattered from beneath the surface and penetrating it (assumed to be Lambertian)
t	=	transmission through the atmosphere of L_{sfc}
t'	=	transmission through the atmosphere of L_{g}

$L_{\rm sa}$ = $L_0 + L_{\rm sfc} t$

$T_{\rm g}$ = the transmission due to the absorptive processes in the gas

$L'_{\rm sa}$ = $L_{\rm sa}/T_{\rm g}$

$t'_{\rm u}$ = effective upward transmission

$t_{\rm u}$ = $t'/T_{\rm g}$ is the upward transmission due to scattering where $t_{\rm u} = t_{\rm u}(\lambda, \tau_{\rm a}, \theta, z_{\rm sen}, z_{\rm sur})$

$t_{\rm d}$ = $t_{\rm d}(\lambda, \tau_{\rm a}, \theta_0)$ is the downward transmittance through the atmosphere to the target

$z_{\rm sen}$ = Sensor altitude

$z_{\rm sur}$ = Surface elevation

μ_0 = $\cos\theta_0$

E_0 = the solar spectral irradiance at the top of the atmosphere

$\rho^*_{\rm obs}$ = observed reflectance; at-sensor apparent reflectance

$\rho^*_{\rm atm+sfc}$ = reflectance due to atmospheric scattering, direct sunlight & diffuse skylight reflection off of the rough ocean surface; this is a tabulated quantity.

$\rho_{\rm g}$ = reflectance of ground target; could be water (assumed Lambertian)

$\rho_{\rm w}$ = the water leaving reflectance (assumed to be Lambertian

$\bar{s}$ = $\bar{s}(\lambda, \tau_{\rm a})$ is the average reflectivity at the base of the atmosphere; that is, the fraction of upwelling light that is reflected back to the base

$R_{\rm rs}$ = remote-sensing reflectance

= $\rho_{\rm g}/\pi$

$[L_{\rm g}]_N$ = normalized ground-leaving radiance reflected of the target (assumed to be Lambertian)

= $E_0 R_{\rm rs}$

N_S = An integer, the number of cross-track samples.

N_L = An integer, the number of along-track lines.

N_B = An integer, the number of bands for each pixel.

N_i = An integer, the scaled representation of the measured radiance at a particular pixel and band in the input image file.

C Parameters for the Tables

The aerosol lookup tables used in the tabular version of *Tafkaa* are stored on the following grid:

- Wavelengths (μm): 0.39, 0.41, 0.44, 0.47,0.51, 0.55, 0.61, 0.67, 0.75, 0.865, 1.04, 1.24, 1.64, 2.25
- View Zenith Angle (degrees): 0.0, 1.5, 6.0, 12.0, 18.0, 24.0, 30.0, 36.0, 42.0, 48.0, 54.0, 60.0, 66.0, 72.0, 78.0, 84.0, 88.0
- Relative Azimuth Angle (degrees): 0.0, 12.0, 24.0, 36.0, 48.0, 60.0, 72.0, 84.0, 90.0, 96.0, 108.0, 120.0, 132.0, 144.0, 156.0, 168.0, 180.0
- Solar Zenith Angle (degrees): 1.5, 12.0, 24.0, 36.0, 48.0, 54.0, 60.0, 66.0, 72.0
- Aerosol optical depth at 0.55 μm: 0.0, 0.1, 0.2, 0.3, 0.5, 0.7, 1.0, 1.3, 1.6, 2.0
- Aerosol Relative Humidity[8] (%): 50, 70, 80, 90, 98
- Aerosol Names[8]: Maritime, Coastal, Coastal-a, Tropospheric, Urban
- Aerosol Table Altitudes (km): 84.0, 22.0, 16.0, 10.5, 8.0, 5.5, 4.0, 2.59, 2.50, 0.

[8] A discussion the aerosol properties may be found in Appendix H.

D The Instrumental Response Function Input File

The typical convolution takes, for example, the transmission on some high-resolution grid (labeled A) to some lower resolution grid (labeled B), such that

$$T_B(\lambda) = \int_{\lambda'_{\rm min}}^{\lambda'_{\rm max}} T_A(\lambda') F(\lambda, \lambda') d\lambda' \ , \tag{11}$$

with the restriction that

$$\int_{\lambda'_{\rm min}}^{\lambda'_{\rm max}} F(\lambda, \lambda') d\lambda' = 1 \ , \tag{12}$$

and assuming that $F(\lambda, \lambda')$ is nonzero only in the range $\lambda'_{\rm min} \leq \lambda' \leq \lambda'_{\rm max}$. In our case, the higher resolution grid has a range of 0.3 – 3.1μm, has a spacing of 0.1 nm, and a FWHM of 0.2 nm; there are a total of 28,001 points in this grid. The instrumental response function that we require relates the observed grid to the aforementioned high resolution grid. For our discretized grids, we may write

$$T_B(\lambda_j) = \sum_{k=k_{\rm min}}^{k_{\rm max}} T_A(k) F(\lambda_j, k) \Delta\lambda'. \tag{13}$$

Thus, the information about the instrumental response function that needs to be tabulated is the product $F(\lambda_j, k)\Delta\lambda'$ for each observed wavelength λ_j. I try to do this using minimal storage space in the manner described below.

The processing of the IRF input file is done by the subroutine `read_instrumental_response_file` in the file `std_to_instr.f90`. The IRF file is opened as a direct access unformatted file, thus it is a binary file with no header records.

It is first opened with a standard real or integer record length (this assumes that default reals and integers have the same length, such as 4 bytes on the machines currently used). The first record is then the typical record length of the file. The file is closed and re-opened with this record length. Because of this, the first record must be padded (usually with zeroes). Thus, if the record length is `nrecl`, the first record consists of the number `nrecl` followed by `nrecl-1` zeroes.

There then follow `nobs` records, one for each of the observed wavelengths. The statement that reads the `nobs` records is

```
do j=1,nobs
   read(unit=file_num,rec=j+1)initial_index(j),ncvtot(j),finstr(:,j)
end do
```

where `inital_index` corresponds to $k_{\rm min}$ above on the internal intermediate grid described above; `ncvtot(j)` is then equal to the number of non-zero elements of the instrumental response function, and as written in the discretized form above `ncvtot(j)`$= k_{\rm max} - k_{\rm min} + 1$. Finally, the instrumental response function is then `finstr(:,j)`$= F(\lambda_j, k)\Delta\lambda'$, where the first value is the first non-zero element. The record should be padded with zeroes to fill the the record up to the standard record length. Thus, the record length should be chosen as $2 + \max(1 + k_{\rm max} - k_{\rm min})$.

E The *Tafkaa* pointing file

When `tafkaa_tabular` is invoked with `tafkaa_geometry = M` and $M = 10$ or 110, `tafkaa_tabular` expects to find a pointing file. The $N_S + 5$ lines in the file must be formatted so that:

1. Five lines of header, that can be anything you want. This should contain useful comments about the file/sensor, where you got these values from, etc.

2. Each following line has one value on it, the average cross-track pointing for each pixel, in degrees.

The code fragment used to read the pointing file is:

```
open(unit=tmp_file_num,&
  file=tafkaa_pointing_file(:len_trim(tafkaa_pointing_file)),&
  access='sequential',action='read',status='old')
do i=1,5 ! read 5 lines of header
  read(tmp_file_num,*)junk
enddo
do i=1,nsamps
  read(tmp_file_num,*)avg_sample_theta(i)
end do
close(tmp_file_num,status='keep')
```

An example of the first few lines of pointing file is:

```
; this is the final pointing file for the 2001 calibration
; of the PHILLS2 data set
line 3
line 4
line 5
17.413906
17.364059
17.314212
17.264364
17.215090
```

⋮

The file continues on as described above.

F The *Tafkaa* line geometry file

When `tafkaa_tabular` is invoked with `tafkaa_geometry = M` and $M = 10$, 100, or 110, `tafkaa_tabular` expects to find a geometry file. The $N_L + 5$ lines of the file must formatted so that:

1. Five lines of header, can be anything you want. Useful for you to put comments about the file/sensor, where you got these values from, etc.
2. Each line has information specifying the date, time, latitude, longitude and heading for at the center of every line.

The Fortran-90 code fragment used to read the line geometry file is:

```
call get_file_unit_number(tmp_file_num)
allocate(lc_lat(nlines),lc_lon(nlines),&
    lc_time(nlines),lc_heading(nlines),&
    lc_second(nlines),lc_year(nlines),&
    lc_month(nlines),lc_date(nlines))
open(unit=tmp_file_num, file=tafkaa_line_geometry_file,&
     access='sequential',action='read', status='old')
do i=1,5 ! read 5 lines of header
   read(tmp_file_num,*)junk
enddo
do i=1,nlines
   read(tmp_file_num,'(I4,4(x,i2),x,F8.5,2(x,f11.6),x,f8.3)')&
   lc_year(i),lc_month(i),lc_date(i),lc_hour(i),lc_minute(i),&
   lc_second(i),lc_lat(i),lc_lon(i),lc_heading(i)
enddo
close(tmp_file_num,status='keep')
```

The year, month, date, hour, minute, second refer to GMT values. The latitude and longitude refer to the coordinates of the central pixel in each line. There is one line in this file per line of the data. Each line contains

- a 4-digit integer year (lc_year) [Gregorian calendar]
- a single space
- a 2-digit integer month (lc_month) [January= 1]
- a single space
- a 2 digit integer date (lc_date) [First day in month is 1]
- a single space
- a 2 digit integer hour (lc_hour) [24 hour clock]
- a single space
- a 2 digit integer minute (lc_minute) [0–59]
- a single space
- a real decimal seconds (lc_second) [≥ 0, < 60.00]
- a single space
- a decimal degrees latitude value (lc_lat), equator = 0°, N> 0°
- a single space
- a decimal degrees longitude value (lc_lon), PM= 0°, E> 0°
- a single space
- a decimal degrees heading value (lc_heading), N = 0°, E = 90°

An example of the first few lines of a line geometry file is:

```
This file prepared by Dave Kohler for July 31 2001 New Jersey Leo-15
For PHILLS2.
Header line 3
Header line 4
Header line 5
2001  7 31 14  9  4.81201   39.322944  -74.598066   76.652
2001  7 31 14  9  4.83948   39.322939  -74.598011   76.665
2001  7 31 14  9  4.90814   39.322930  -74.597899   76.694
2001  7 31 14  9  4.96307   39.322922  -74.597785   76.727
2001  7 31 14  9  5.03174   39.322914  -74.597672   76.763
```

While *Tafkaa* requires that there be one line in the geometry file for each line in the data cube, it does not require that the lines be unique. In many instances, the frame rate is so fast (and the plane is slow enough) that repeated latitude, longitude, and time values should not matter. In fact, AVIRIS navigation files update this information only once a second. This should be sufficient.

G Example Header Files

In this section we include full examples of input files, input image header files, and output header files. About the only difference between the files as shown here and the files as they exist on the computer is the necessary existence of page breaks when the files are displayed as I have chosen to display them here. There are no page breaks in the input files on the computer.

G.1 Input Files

G.1.1 *Tafkaa* Input File

```
tafkaa_input_image_name = ./coral.bip
tafkaa_data_directory = ../
mask_image_name = ./coral_test_mask.img
tafkaa_use_which_masks = { land }
tafkaa_output_root_name = ./coral_test
tafkaa_output_type = refl
tafkaa_output_scale_factor = 10000
tafkaa_ground_elevation = 0.00
tafkaa_atmo_model = tropical
tafkaa_atmo_gasses = {H2O, CO2, O3, N2O, CO, CH4, O2}
tafkaa_atmo_ozone = 0.34
tafkaa_use_prev_atmo_trans = 1
tafkaa_prev_atmo_trans_file = ./coral_test_tafkaa.vapbin
tafkaa_h2o_enter_inputs = 1
tafkaa_h2o_wl_set1 = {0.8650, 1.030, 0.945}
tafkaa_h2o_nb_set1 = {3, 3, 5}
tafkaa_h2o_wl_set2 = {1.0650, 1.240, 1.140}
tafkaa_h2o_nb_set2 = {3, 3, 5}
tafkaa_aerosol_method = 0
tafkaa_aerosol_rh = 80%
tafkaa_aerosol_model = coastal-a
tafkaa_aerosol_tau550 = 0.1
tafkaa_aerosol_weights = {0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,1.,1.,1.,1.}
tafkaa_wind_speed = 2
```

G.1.2 Input Radiance Image Header File

Based on the input file in Appendix G.1.1, the name of the header file below should be `./coral.hdr`. On Unix-style operating systems, however, if that name is not found, *Tafkaa* will also search for the name `./coral.bip.hdr`. As can be seen from the history portion of the file below, it was generated on PC using W. Snyder's header population widget from his ENVI® interface routines. Long lines have been re-formatted to typeset correctly on these pages. Recall that the maximum allowed line length is 132 characters.

```
ENVI
description = {
File Resize Result, x resize factor: 1.0000, y resize factor: 1.0000. [Thu
Sep 11 17:21:43 1997]}
samples = 614
lines = 600
bands = 224
header offset = 0
interleave = bip
data type = 2
byte order = 1
file type = ENVI Standard
sensor type = AVIRIS
y start = 298
image_unscaled_units = W/m2/micron/ster
image_center_date = {1996, 3, 23}
image_center_time = {19, 44,   29.000}
image_center_long = {81, 47,   54.000}
image_center_long_hem = W
```

```
image_center_lat = {24, 36,   44.000}
image_center_lat_hem = N
image_center_zenith_ang = {0, 0,    0.000}
image_center_azimuth_ang = {0, 0,    0.000}
sensor_altitude = 19.768000
image_scale_factor = { 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100}
wavelength = { 0.369850, 0.379690, 0.389530, 0.399370, 0.409210, 0.419060, 0.428910,
0.438760, 0.448610, 0.458460, 0.468310, 0.478170, 0.488020, 0.497880, 0.507740, 0.517600,
0.527470, 0.537330, 0.547200, 0.557070, 0.566940, 0.576810, 0.586680, 0.596560, 0.606430,
0.616310, 0.626190, 0.636070, 0.645960, 0.655840, 0.665730, 0.675620, 0.664140, 0.673720,
0.683300, 0.692880, 0.702460, 0.712040, 0.721630, 0.731210, 0.740800, 0.750380, 0.759970,
0.769560, 0.779140, 0.788730, 0.798320, 0.807910, 0.817500, 0.827090, 0.836680, 0.846280,
0.855870, 0.865470, 0.875060, 0.884660, 0.894250, 0.903850, 0.913450, 0.923050, 0.932640,
0.942240, 0.951850, 0.961450, 0.971050, 0.980650, 0.990250, 0.999860, 1.009460, 1.019070,
1.028680, 1.038280, 1.047890, 1.057500, 1.067110, 1.076720, 1.086330, 1.095940, 1.105550,
1.115160, 1.124780, 1.134390, 1.144010, 1.153620, 1.163240, 1.172860, 1.182470, 1.192090,
1.201710, 1.211330, 1.220950, 1.230570, 1.240190, 1.249820, 1.259440, 1.269060, 1.254320,
1.264280, 1.274240, 1.284200, 1.294150, 1.304110, 1.314070, 1.324030, 1.333990, 1.343950,
1.353910, 1.363870, 1.373820, 1.383780, 1.393740, 1.403700, 1.413660, 1.423610, 1.433570,
1.443530, 1.453490, 1.463440, 1.473400, 1.483360, 1.493310, 1.503270, 1.513230, 1.523180,
1.533140, 1.543100, 1.553050, 1.563010, 1.572970, 1.582920, 1.592880, 1.602830, 1.612790,
1.622740, 1.632700, 1.642650, 1.652610, 1.662560, 1.672520, 1.682470, 1.692430, 1.702380,
1.712340, 1.722290, 1.732250, 1.742200, 1.752150, 1.762110, 1.772060, 1.782020, 1.791970,
1.801920, 1.811880, 1.821830, 1.831780, 1.841730, 1.851690, 1.861640, 1.871590, 1.881540,
1.878300, 1.888340, 1.898390, 1.908430, 1.918460, 1.928500, 1.938530, 1.948560, 1.958590,
1.968620, 1.978640, 1.988670, 1.998690, 2.008700, 2.018720, 2.028730, 2.038740, 2.048750,
2.058760, 2.068760, 2.078770, 2.088770, 2.098770, 2.108760, 2.118760, 2.128750, 2.138740,
2.148720, 2.158710, 2.168690, 2.178670, 2.188650, 2.198630, 2.208600, 2.218570, 2.228540,
2.238510, 2.248480, 2.258440, 2.268400, 2.278360, 2.288320, 2.298270, 2.308220, 2.318170,
2.328120, 2.338070, 2.348010, 2.357950, 2.367890, 2.377830, 2.387760, 2.397690, 2.407620,
2.417550, 2.427480, 2.437400, 2.447320, 2.457240, 2.467160, 2.477080, 2.486990, 2.496900,
2.506810}
fwhm = { 0.009610, 0.009580, 0.009550, 0.009530, 0.009500, 0.009480, 0.009460, 0.009440,
0.009420, 0.009400, 0.009380, 0.009370, 0.009350, 0.009340, 0.009330, 0.009310, 0.009300,
0.009290, 0.009280, 0.009280, 0.009270, 0.009260, 0.009260, 0.009260, 0.009250, 0.009250,
0.009250, 0.009250, 0.009260, 0.009260, 0.009260, 0.009270, 0.008300, 0.008310, 0.008330,
0.008340, 0.008350, 0.008360, 0.008370, 0.008390, 0.008400, 0.008410, 0.008420, 0.008430,
0.008440, 0.008450, 0.008460, 0.008460, 0.008470, 0.008480, 0.008490, 0.008500, 0.008500,
0.008510, 0.008520, 0.008530, 0.008530, 0.008540, 0.008540, 0.008550, 0.008550, 0.008560,
0.008560, 0.008570, 0.008570, 0.008570, 0.008580, 0.008580, 0.008580, 0.008580, 0.008590,
0.008590, 0.008590, 0.008590, 0.008590, 0.008590, 0.008590, 0.008590, 0.008590, 0.008590,
0.008590, 0.008590, 0.008590, 0.008590, 0.008580, 0.008580, 0.008580, 0.008580, 0.008570,
0.008570, 0.008570, 0.008560, 0.008560, 0.008550, 0.008550, 0.008540, 0.010860, 0.010870,
```

```
0.010880, 0.010890, 0.010890, 0.010900, 0.010900, 0.010910, 0.010910, 0.010910, 0.010920,
0.010920, 0.010920, 0.010920, 0.010920, 0.010920, 0.010920, 0.010920, 0.010920, 0.010910,
0.010910, 0.010910, 0.010900, 0.010900, 0.010890, 0.010880, 0.010880, 0.010870, 0.010860,
0.010850, 0.010840, 0.010830, 0.010820, 0.010810, 0.010800, 0.010790, 0.010770, 0.010760,
0.010750, 0.010730, 0.010720, 0.010700, 0.010690, 0.010670, 0.010650, 0.010630, 0.010610,
0.010590, 0.010570, 0.010550, 0.010530, 0.010510, 0.010490, 0.010470, 0.010440, 0.010420,
0.010390, 0.010370, 0.010340, 0.010320, 0.010290, 0.010260, 0.010230, 0.010210, 0.011180,
0.011160, 0.011140, 0.011130, 0.011110, 0.011090, 0.011080, 0.011060, 0.011040, 0.011020,
0.011010, 0.010990, 0.010970, 0.010950, 0.010940, 0.010920, 0.010900, 0.010880, 0.010870,
0.010850, 0.010830, 0.010810, 0.010800, 0.010780, 0.010760, 0.010740, 0.010720, 0.010710,
0.010690, 0.010670, 0.010650, 0.010630, 0.010620, 0.010600, 0.010580, 0.010560, 0.010540,
0.010520, 0.010500, 0.010490, 0.010470, 0.010450, 0.010430, 0.010410, 0.010390, 0.010370,
0.010350, 0.010340, 0.010320, 0.010300, 0.010280, 0.010260, 0.010240, 0.010220, 0.010200,
0.010180, 0.010160, 0.010140, 0.010120, 0.010110, 0.010090, 0.010070, 0.010050, 0.010030}

history = begins

nemo_header_test, version 1.0
Thu Oct 14 16:01:04 1999
sensor type = AVIRIS
cal_file = D:\new_data_system\examples\coral_cal_file.asc
image_center_date = { 1996, 3, 23}
image_center_time = { 19, 44, 29.0000}
image_center_long = { 81, 47, 54.0000}
image_center_long_hem = W
image_center_lat = { 24, 36, 44.0000}
image_center_lat_hem = N
image_center_zenith_ang = { 0, 0, 0.000000}
image_center_azimuth_ang = { 0, 0, 0.000000}
sensor_altitude =  19.7680

history = ends
```

G.2 Output Header Files

G.2.1 *Tafkaa* Output Header File

Based on the above files, the name of the *Tafkaa* output image will be `./coral_test_tafkaa_refl.img` and the image header file will be `./coral_test_tafkaa_refl.hdr`. This output header file has been formatted to better fit the page; additionally, some of the elelements of the `band names` keyword have been removed since that information is not very important in understanding the output file.

```
ENVI
description = {
  Image data cube: surface  reflectance (dimensionless)x10000.0
  derived from tafkaa (tabular);
  measurement date: 1996-03-23; measurement time: 19:44:29.000 GMT
  latitude:  24deg 36m 44.000s N ; longitude:  81deg 47m 54.000s W
  view zenith angle:   0deg  0m  0.000s ;
  relative azimuth angle:   0deg  0m  0.000s ;
  atmospheric model: tropical           ;
  absorption includes : H_2O CO_2 O_3 N_2O CO CH_4 O_2;
  ozone amount:  0.340 atm-cm ;
  windspeed:   2.00 m/s;
  weights for determining aerosols:
```

```
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 1.00 1.00 1.00 }
samples = 614
lines   = 600
bands   = 224
header offset = 0
file type = ENVI Standard
data type = 2
interleave = bip
byte order = 1
sensor type = AVIRIS
image_center_date = { 1996, 03, 23}
image_center_time = { 19., 44., 29.000}
image_center_lat = { 24., 36., 44.000}
image_center_lat_hem = N
image_center_long = {  81., 47., 54.000}
image_center_long_hem = W
image_center_zenith_ang = {  0.,  0.,  0.000}
image_center_azimuth_ang = {   0.,  0.,  0.000}
sensor_altitude =    19.768
image_scale_factor = {   10000.0000}
image_unscaled_units = { dimensionless}
wavelength = {
 0.36985, 0.37969, 0.38953, 0.39937, 0.40921, 0.41906, 0.42891, 0.43876, 0.44861,
 0.45846, 0.46831, 0.47817, 0.48802, 0.49788, 0.50774, 0.51760, 0.52747, 0.53733,
 0.54720, 0.55707, 0.56694, 0.57681, 0.58668, 0.59656, 0.60643, 0.61631, 0.62619,
 0.63607, 0.64596, 0.65584, 0.66573, 0.67562, 0.66414, 0.67372, 0.68330, 0.69288,
 0.70246, 0.71204, 0.72163, 0.73121, 0.74080, 0.75038, 0.75997, 0.76956, 0.77914,
 0.78873, 0.79832, 0.80791, 0.81750, 0.82709, 0.83668, 0.84628, 0.85587, 0.86547,
 0.87506, 0.88466, 0.89425, 0.90385, 0.91345, 0.92305, 0.93264, 0.94224, 0.95185,
 0.96145, 0.97105, 0.98065, 0.99025, 0.99986, 1.00946, 1.01907, 1.02868, 1.03828,
 1.04789, 1.05750, 1.06711, 1.07672, 1.08633, 1.09594, 1.10555, 1.11516, 1.12478,
 1.13439, 1.14401, 1.15362, 1.16324, 1.17286, 1.18247, 1.19209, 1.20171, 1.21133,
 1.22095, 1.23057, 1.24019, 1.24982, 1.25944, 1.26906, 1.25432, 1.26428, 1.27424,
 1.28420, 1.29415, 1.30411, 1.31407, 1.32403, 1.33399, 1.34395, 1.35391, 1.36387,
 1.37382, 1.38378, 1.39374, 1.40370, 1.41366, 1.42361, 1.43357, 1.44353, 1.45349,
 1.46344, 1.47340, 1.48336, 1.49331, 1.50327, 1.51323, 1.52318, 1.53314, 1.54310,
 1.55305, 1.56301, 1.57297, 1.58292, 1.59288, 1.60283, 1.61279, 1.62274, 1.63270,
 1.64265, 1.65261, 1.66256, 1.67252, 1.68247, 1.69243, 1.70238, 1.71234, 1.72229,
 1.73225, 1.74220, 1.75215, 1.76211, 1.77206, 1.78202, 1.79197, 1.80192, 1.81188,
 1.82183, 1.83178, 1.84173, 1.85169, 1.86164, 1.87159, 1.88154, 1.87830, 1.88834,
 1.89839, 1.90843, 1.91846, 1.92850, 1.93853, 1.94856, 1.95859, 1.96862, 1.97864,
 1.98867, 1.99869, 2.00870, 2.01872, 2.02873, 2.03874, 2.04875, 2.05876, 2.06876,
 2.07877, 2.08877, 2.09877, 2.10876, 2.11876, 2.12875, 2.13874, 2.14872, 2.15871,
 2.16869, 2.17867, 2.18865, 2.19863, 2.20860, 2.21857, 2.22854, 2.23851, 2.24848,
 2.25844, 2.26840, 2.27836, 2.28832, 2.29827, 2.30822, 2.31817, 2.32812, 2.33807,
 2.34801, 2.35795, 2.36789, 2.37783, 2.38776, 2.39769, 2.40762, 2.41755, 2.42748,
 2.43740, 2.44732, 2.45724, 2.46716, 2.47708, 2.48699, 2.49690, 2.50681}
fwhm = {
 0.00961, 0.00958, 0.00955, 0.00953, 0.00950, 0.00948, 0.00946, 0.00944, 0.00942,
 0.00940, 0.00938, 0.00937, 0.00935, 0.00934, 0.00933, 0.00931, 0.00930, 0.00929,
 0.00928, 0.00928, 0.00927, 0.00926, 0.00926, 0.00926, 0.00925, 0.00925, 0.00925,
 0.00925, 0.00926, 0.00926, 0.00926, 0.00927, 0.00830, 0.00831, 0.00833, 0.00834,
 0.00835, 0.00836, 0.00837, 0.00839, 0.00840, 0.00841, 0.00842, 0.00843, 0.00844,
 0.00845, 0.00846, 0.00846, 0.00847, 0.00848, 0.00849, 0.00850, 0.00850, 0.00851,
```

```
 0.00852, 0.00853, 0.00853, 0.00854, 0.00854, 0.00855, 0.00855, 0.00856, 0.00856,
 0.00857, 0.00857, 0.00857, 0.00858, 0.00858, 0.00858, 0.00858, 0.00859, 0.00859,
 0.00859, 0.00859, 0.00859, 0.00859, 0.00859, 0.00859, 0.00859, 0.00859, 0.00859,
 0.00859, 0.00859, 0.00859, 0.00858, 0.00858, 0.00858, 0.00858, 0.00857, 0.00857,
 0.00857, 0.00856, 0.00856, 0.00855, 0.00855, 0.00854, 0.01086, 0.01087, 0.01088,
 0.01089, 0.01089, 0.01090, 0.01090, 0.01091, 0.01091, 0.01091, 0.01092, 0.01092,
 0.01092, 0.01092, 0.01092, 0.01092, 0.01092, 0.01092, 0.01092, 0.01091, 0.01091,
 0.01091, 0.01090, 0.01090, 0.01089, 0.01088, 0.01088, 0.01087, 0.01086, 0.01085,
 0.01084, 0.01083, 0.01082, 0.01081, 0.01080, 0.01079, 0.01077, 0.01076, 0.01075,
 0.01073, 0.01072, 0.01070, 0.01069, 0.01067, 0.01065, 0.01063, 0.01061, 0.01059,
 0.01057, 0.01055, 0.01053, 0.01051, 0.01049, 0.01047, 0.01044, 0.01042, 0.01039,
 0.01037, 0.01034, 0.01032, 0.01029, 0.01026, 0.01023, 0.01021, 0.01118, 0.01116,
 0.01114, 0.01113, 0.01111, 0.01109, 0.01108, 0.01106, 0.01104, 0.01102, 0.01101,
 0.01099, 0.01097, 0.01095, 0.01094, 0.01092, 0.01090, 0.01088, 0.01087, 0.01085,
 0.01083, 0.01081, 0.01080, 0.01078, 0.01076, 0.01074, 0.01072, 0.01071, 0.01069,
 0.01067, 0.01065, 0.01063, 0.01062, 0.01060, 0.01058, 0.01056, 0.01054, 0.01052,
 0.01050, 0.01049, 0.01047, 0.01045, 0.01043, 0.01041, 0.01039, 0.01037, 0.01035,
 0.01034, 0.01032, 0.01030, 0.01028, 0.01026, 0.01024, 0.01022, 0.01020, 0.01018,
 0.01016, 0.01014, 0.01012, 0.01011, 0.01009, 0.01007, 0.01005, 0.01003}
bbl = {
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
band names = {
 refl[0.370 micron] x 10000.0, refl[0.380 micron] x 10000.0,
 refl[0.390 micron] x 10000.0, refl[0.399 micron] x 10000.0,
 refl[0.409 micron] x 10000.0, refl[0.419 micron] x 10000.0,
 refl[0.429 micron] x 10000.0, refl[0.439 micron] x 10000.0,
```

⋮

This continues on for all the bands in the image, so we'll pick up near the end of the `band names` array:

⋮

```
 refl[2.457 micron] x 10000.0, refl[2.467 micron] x 10000.0,
 refl[2.477 micron] x 10000.0, refl[2.487 micron] x 10000.0,
 refl[2.497 micron] x 10000.0, refl[2.507 micron] x 10000.0}

history = begins

nemo_header_test, version 1.0
Thu Oct 14 16:01:04 1999
sensor type = AVIRIS
cal_file = D:\new_data_system\examples\coral_cal_file.asc
image_center_date = { 1996, 3, 23}
image_center_time = { 19, 44, 29.0000}
image_center_long = { 81, 47, 54.0000}
image_center_long_hem = W
image_center_lat = { 24, 36, 44.0000}
image_center_lat_hem = N
```

```
image_center_zenith_ang = { 0, 0, 0.000000}
image_center_azimuth_ang = { 0, 0, 0.000000}
sensor_altitude =  19.7680

mask_name = mask for SGI
mask_version = 0.9i 1999-Oct-21-16:25:00 EDT
mask_execution_date = 1999-Oct-26
mask_execution_time = 10:31:40   -0500
mask_input_image_name = ./coral.bip
mask_which_masks = { land, cirrus, low cloud}
mask_data_directory = ../
mask_output_root_name = ./coral_test
ndvi_scale_factor =  10000.0000
land_mask_threshold =      0.0500
cirrus_mask_threshold =      0.0045
cloud_mask_ocean_threshold =      0.1000

tafkaa_name = tAFKaA (tabular) for SGI
tafkaa_version = 0.9i 1999-Nov-09-16:24:00 EDT
tafkaa_execution_date = 1999-Nov-10
tafkaa_execution_time = 17:00:20   -0500
tafkaa_input_image_name = ./coral.bip
tafkaa_ground_elevation =  0.000
tafkaa_atmo_model = tropical
tafkaa_atmo_gasses = {H2O,CO2,O3 ,N2O,CO ,CH4,O2 }
tafkaa_atmo_ozone = 0.340
tafkaa_h2o_enter_inputs = 1
tafkaa_h2o_wl_set1 = { 0.8650, 1.0300, 0.9450}
tafkaa_h2o_nb_set1 = {  3,  3,  5}
tafkaa_h2o_wl_set2 = { 1.0650, 1.2400, 1.1400}
tafkaa_h2o_nb_set2 = {  3,  3,  5}
tafkaa_wind_speed =  2.00
tafkaa_aerosol_method =       0
tafkaa_aerosol_model = coastal-a
tafkaa_aerosol_rh = 80%
tafkaa_aerosol_tau550 = 0.100
tafkaa_aerosol_weights = { 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.}
tafkaa_use_prev_atmo_trans = 1
tafkaa_prev_atmo_trans_file = ./coral_test_tafkaa.vapbin
tafkaa_data_directory = ../
mask_image_name = ./coral_test_mask.img
tafkaa_use_which_masks = { land }
tafkaa_output_type = refl
tafkaa_output_root_name = ./coral_test
tafkaa_output_scale_factor =  10000.0000

history = ends
```

G.2.2 Product Output Header File

Based on the above files, the name of the mask output image will be `./coral_test_tafkaa_prod.img` and the image header file will be `./coral_test_tafkaa_prod.hdr`.

```
ENVI
description = {
```

```
  Product file
  derived from tafkaa (tabular);
  measurement date: 1996-03-23; measurement time: 19:44:29.000 GMT
  latitude:  24deg 36m 44.000s N ; longitude:  81deg 47m 54.000s W
  view zenith angle:   0deg  0m  0.000s ;
  relative azimuth angle:   0deg  0m  0.000s ;
  atmospheric model: tropical           ;
  absorption includes : H_2O CO_2 O_3 N_2O CO CH_4 O_2;
  ozone amount:  0.340 atm-cm ;
  windspeed:   2.00 m/s;
  weights for determining aerosols:
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 1.00 1.00 1.00 }
samples = 614
lines   = 600
bands   = 5
header offset = 0
file type = ENVI Standard
data type = 2
interleave = bsq
byte order = 1
sensor type = AVIRIS
image_center_date = { 1996, 03, 23}
image_center_time = { 19., 44., 29.000}
image_center_lat = { 24., 36., 44.000}
image_center_lat_hem = N
image_center_long = {  81., 47., 54.000}
image_center_long_hem = W
image_center_zenith_ang = {  0.,  0.,  0.000}
image_center_azimuth_ang = {   0.,  0.,  0.000}
sensor_altitude =    19.768
image_scale_factor = {
   1000.0000,   1000.0000,      1.0000,      1.0000,    100.0000}
image_unscaled_units = {  cm of H2O, tauaer550, RH %, none, none}
band names = {
1000.000x cm of H2O,1000.000x tau aerosol (550nm), %RH, model, 100.000xfitqual}

history = begins

nemo_header_test, version 1.0
Thu Oct 14 16:01:04 1999
sensor type = AVIRIS
cal_file = D:\new_data_system\examples\coral_cal_file.asc
image_center_date = { 1996, 3, 23}
image_center_time = { 19, 44, 29.0000}
image_center_long = { 81, 47, 54.0000}
image_center_long_hem = W
image_center_lat = { 24, 36, 44.0000}
image_center_lat_hem = N
image_center_zenith_ang = { 0, 0, 0.000000}
image_center_azimuth_ang = { 0, 0, 0.000000}
sensor_altitude =  19.7680

mask_name = mask for SGI
mask_version = 0.9i 1999-Oct-21-16:25:00 EDT
```

```
mask_execution_date = 1999-Oct-26
mask_execution_time = 10:31:40   -0500
mask_input_image_name = ./coral.bip
mask_which_masks = { land, cirrus, low cloud}
mask_data_directory = ../
mask_output_root_name = ./coral_test
ndvi_scale_factor =  10000.0000
land_mask_threshold =      0.0500
cirrus_mask_threshold =      0.0045
cloud_mask_ocean_threshold =      0.1000

tafkaa_name = tAFKaA (tabular) for SGI
tafkaa_version = 0.9i 1999-Nov-09-16:24:00 EDT
tafkaa_execution_date = 1999-Nov-10
tafkaa_execution_time = 17:00:20   -0500
tafkaa_input_image_name = ./coral.bip
tafkaa_ground_elevation =  0.000
tafkaa_atmo_model = tropical
tafkaa_atmo_gasses = {H2O,CO2,O3 ,N2O,CO ,CH4,O2 }
tafkaa_atmo_ozone = 0.340
tafkaa_h2o_enter_inputs = 1
tafkaa_h2o_wl_set1 = { 0.8650, 1.0300, 0.9450}
tafkaa_h2o_nb_set1 = {  3,  3,  5}
tafkaa_h2o_wl_set2 = { 1.0650, 1.2400, 1.1400}
tafkaa_h2o_nb_set2 = {  3,  3,  5}
tafkaa_wind_speed =  2.00
tafkaa_aerosol_method =       0
tafkaa_aerosol_model = coastal-a
tafkaa_aerosol_rh = 80%
tafkaa_aerosol_tau550 = 0.100
tafkaa_aerosol_weights = { 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.}
tafkaa_use_prev_atmo_trans = 1
tafkaa_prev_atmo_trans_file = ./coral_test_tafkaa.vapbin
tafkaa_data_directory = ../
mask_image_name = ./coral_test_mask.img
tafkaa_use_which_masks = { land }
tafkaa_output_type = refl
tafkaa_output_root_name = ./coral_test
tafkaa_output_scale_factor =  10000.0000

history = ends
```

H Aerosol Properties

H.1 Aerosol Properties for the 6S version

Comments	Model Name		
	C	M	U
Dust like, Continental origin	0.70		0.17
Water-soluble	0.29	0.05	0.61
Oceanic origin, sea salt solution in water		0.95	
Soot	0.01		0.22

Table 1: A rough description of the types of aerosols present, and the fractions present in the different aerosol models used in the 6S version of *Tafkaa*. The abbreviations used for the model names are: C (Continental); M (Maritime); and U (Urban).

The aerosols for the 6S version of *Tafkaa* all assume 70% relative humidity. The optical parameters of these components, as well as the specified ratios, as from WMO [1986].

H.2 Aerosol Properties for the tabular version

Mode	Comments	Model Name				
		M	C	C-a	T	U
1	Continental origin rural aerosol mixture	0.990	0.995	0.998	1.000	
2	Oceanic origin, sea salt solution in water	0.010	0.005	0.002		
3	Urban aerosol mixture, rural & soot					$1 - 1.25\text{e-}4$
4	Urban aerosol mixture, rural & soot					1.25e-4

Table 2: A rough description of the types of aerosols present, and the fractions present in the different aerosol models used in the tabular version of *Tafkaa*. The abbreviations used for the model names are: M (Maritime); C (Coastal); C-a (Coastal-a); T (Tropospheric); and U (Urban).

All the aerosol distributions used are log-normal; most are bi-modal[9] . Component i is represented as

$$\frac{dN_i(r)}{dr} = n_i(r) = \frac{N f_i}{r\sigma_i\sqrt{2\pi}} \exp\left\{-\frac{(\ln r - \ln r_{m_i})^2}{2\sigma_i^2}\right\} . \tag{14}$$

Here, N is the total number density of particles, f_i is the fractional number density of species i, r is the particle radius, σ_i is the standard deviation of $\ln r$, i.e., $\sigma_i^2 = \langle(\ln r - \ln r_{m_i})^2\rangle$, and r_{m_i} is the mean radius of $\ln r$. Then, the total particle distribution is described as the sum over K components as

$$\frac{dN(r)}{dr} = n(r) = \sum_{i=1}^{K} \frac{dN_i(r)}{dr} . \tag{15}$$

The properties used in determining the aerosols come from Shettle and Fenn [1979]. Currently four different aerosol modes are used, and they are used to make 5 general aerosol models. The proportions of each mode in the different aerosol models is given in Table 2, the parameters of their log-normal distributions are given in Table 3, and the frequency dependent aerosol optical properties are listed in Tables 4 – 8.

The aerosol information listed in these tables is read by Zia Ahmad's program `phs`[10] which creates phase function files for the various aerosols. His `rt1` or `rt1.crft` then applies a distribution of the aerosols throughout the atmosphere in order to get reasonable distributions. Finally, `rt2` or `rt2.crft` perform the radiative transfer calculations using output from both of the above, and create the files that are later parsed into the lookup tables used by *Tafkaa*.

[9] The tropospheric model is unimodal, as seen in Table 2.

[10] The format is different than is listed here. The format of the tables in this document is more concise.

Mode	r_m (micron) RH=50%	RH=70%	RH=80%	RH=90%	RH=98%	σ
1	2.748e-02	2.846e-02	3.274e-02	3.884e-02	4.751e-02	8.059e-01
2	1.711e-01	2.041e-01	3.180e-01	3.803e-01	6.024e-01	9.210e-01
3	2.563e-02	2.911e-02	3.514e-02	4.187e-02	5.996e-02	8.059e-01
4	4.113e-01	4.777e-01	5.805e-01	7.061e-01	1.169e-00	9.210e-01

Table 3: $r_{m_i} = \exp(\langle \ln r \rangle)$ (columns 2 – 6) and $\sigma_i = \sqrt{\langle (\ln r - \ln r_{m_i})^2 \rangle}$ for different aerosol modes. σ_i does not depend on relative humidity.

λ	RH = 50% Mode 1		Mode 2		Mode 3		Mode 4	
(μm)	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$
0.390	1.520	5.60e-3	1.472	3.75e-8	1.557	9.01e-2	1.555	8.93e-2
0.410	1.520	5.60e-3	1.471	2.37e-8	1.557	8.95e-2	1.555	8.88e-2
0.440	1.520	5.60e-3	1.470	2.07e-8	1.557	8.89e-2	1.555	8.81e-2
0.470	1.520	5.60e-3	1.470	1.80e-8	1.557	8.83e-2	1.555	8.75e-2
0.510	1.520	5.60e-3	1.470	9.42e-9	1.557	8.79e-2	1.555	8.71e-2
0.550	1.520	6.26e-3	1.470	8.54e-9	1.557	8.66e-2	1.555	8.58e-2
0.610	1.520	6.26e-3	1.464	1.53e-8	1.557	8.52e-2	1.555	8.45e-2
0.670	1.520	6.65e-3	1.461	4.78e-8	1.557	8.50e-2	1.555	8.43e-2
0.750	1.517	7.90e-3	1.458	2.70e-7	1.554	8.61e-2	1.552	8.53e-2
0.865	1.510	1.03e-2	1.452	2.79e-6	1.549	8.79e-2	1.547	8.72e-2
1.040	1.510	1.32e-2	1.445	1.08e-4	1.549	9.18e-2	1.547	9.10e-2
1.240	1.492	1.50e-2	1.443	2.80e-4	1.536	9.48e-2	1.534	9.40e-2
1.640	1.448	1.59e-2	1.430	5.69e-4	1.505	9.94e-2	1.503	9.85e-2
2.250	1.356	9.22e-3	1.413	1.71e-3	1.439	1.00e-1	1.437	9.92e-2

Table 4: Optical properties for the aerosol modes for RH = 50%.

λ	RH = 70% Mode 1		Mode 2		Mode 3		Mode 4	
(μm)	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$
0.390	1.502	5.04e-3	1.418	2.31e-8	1.488	6.15e-2	1.477	5.70e-2
0.410	1.502	5.04e-3	1.416	1.47e-8	1.488	6.11e-2	1.477	5.66e-2
0.440	1.502	5.04e-3	1.416	1.27e-8	1.487	6.07e-2	1.476	5.62e-2
0.470	1.502	5.04e-3	1.415	1.11e-8	1.487	6.03e-2	1.476	5.58e-2
0.510	1.501	5.04e-3	1.414	6.04e-9	1.486	6.00e-2	1.475	5.56e-2
0.550	1.501	5.64e-3	1.413	5.84e-9	1.486	5.91e-2	1.474	5.48e-2
0.610	1.501	5.64e-3	1.410	1.28e-8	1.485	5.82e-2	1.474	5.39e-2
0.670	1.501	5.99e-3	1.408	3.81e-8	1.485	5.80e-2	1.474	5.38e-2
0.750	1.498	7.11e-3	1.406	1.89e-7	1.483	5.88e-2	1.472	5.45e-2
0.865	1.492	9.29e-3	1.402	1.79e-6	1.479	6.00e-2	1.468	5.56e-2
1.040	1.492	1.19e-2	1.396	6.53e-5	1.478	6.27e-2	1.467	5.81e-2
1.240	1.475	1.35e-2	1.394	1.75e-4	1.469	6.47e-2	1.458	6.00e-2
1.640	1.435	1.43e-2	1.383	3.79e-4	1.445	6.78e-2	1.435	6.29e-2
2.250	1.350	8.34e-3	1.363	1.17e-3	1.392	6.84e-2	1.385	6.34e-2

Table 5: Optical properties for the aerosol modes for RH = 70%.

λ	RH = 80%							
	Mode 1		Mode 2		Mode 3		Mode 4	
(μm)	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$
0.390	1.447	3.31e-3	1.361	7.94e-9	1.424	3.49e-2	1.417	3.17e-2
0.410	1.446	3.31e-3	1.359	5.16e-9	1.423	3.47e-2	1.416	3.16e-2
0.440	1.445	3.31e-3	1.358	4.40e-9	1.422	3.45e-2	1.415	3.13e-2
0.470	1.445	3.31e-3	1.357	3.75e-9	1.422	3.43e-2	1.414	3.11e-2
0.510	1.444	3.31e-3	1.355	2.46e-9	1.421	3.41e-2	1.413	3.10e-2
0.550	1.443	3.70e-3	1.354	3.00e-9	1.420	3.36e-2	1.412	3.05e-2
0.610	1.443	3.70e-3	1.353	9.76e-9	1.419	3.31e-2	1.411	3.00e-2
0.670	1.443	3.93e-3	1.352	2.71e-8	1.419	3.30e-2	1.411	3.00e-2
0.750	1.440	4.67e-3	1.350	1.02e-7	1.417	3.34e-2	1.409	3.04e-2
0.865	1.436	6.10e-3	1.348	7.35e-7	1.414	3.41e-2	1.406	3.10e-2
1.040	1.435	7.80e-3	1.345	2.00e-5	1.413	3.56e-2	1.405	3.24e-2
1.240	1.423	8.90e-3	1.342	6.40e-5	1.406	3.68e-2	1.399	3.35e-2
1.640	1.394	9.42e-3	1.334	1.78e-4	1.389	3.86e-2	1.382	3.51e-2
2.250	1.330	5.61e-3	1.311	5.95e-4	1.349	3.91e-2	1.344	3.55e-2

Table 6: Optical properties for the aerosol modes for RH = 80%.

λ	RH = 90%							
	Mode 1		Mode 2		Mode 3		Mode 4	
(μm)	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$
0.390	1.404	1.98e-3	1.352	5.66e-9	1.390	2.07e-2	1.383	1.76e-2
0.410	1.403	1.98e-3	1.351	3.74e-9	1.389	2.05e-2	1.381	1.75e-2
0.440	1.402	1.98e-3	1.349	3.15e-9	1.388	2.04e-2	1.380	1.74e-2
0.470	1.401	1.98e-3	1.348	2.66e-9	1.387	2.02e-2	1.379	1.73e-2
0.510	1.400	1.98e-3	1.347	1.92e-9	1.385	2.02e-2	1.378	1.72e-2
0.550	1.399	2.22e-3	1.345	2.58e-9	1.384	1.99e-2	1.377	1.70e-2
0.610	1.399	2.22e-3	1.344	9.24e-9	1.384	1.95e-2	1.376	1.67e-2
0.670	1.398	2.36e-3	1.343	2.53e-8	1.383	1.95e-2	1.375	1.67e-2
0.750	1.396	2.80e-3	1.342	8.83e-8	1.382	1.97e-2	1.374	1.69e-2
0.865	1.393	3.65e-3	1.340	5.77e-7	1.379	2.02e-2	1.372	1.72e-2
1.040	1.391	4.67e-3	1.337	1.31e-5	1.377	2.11e-2	1.370	1.80e-2
1.240	1.383	5.34e-3	1.335	4.71e-5	1.372	2.18e-2	1.365	1.86e-2
1.640	1.362	5.69e-3	1.326	1.48e-4	1.359	2.29e-2	1.353	1.95e-2
2.250	1.315	3.52e-3	1.303	5.10e-4	1.326	2.32e-2	1.321	1.99e-2

Table 7: Optical properties for the aerosol modes for RH = 90%.

λ	RH = 98%							
	Mode 1		Mode 2		Mode 3		Mode 4	
(μm)	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$	$\mathcal{R}(n)$	$\mathcal{I}(n)$
0.390	1.375	1.08e-3	1.343	3.22e-9	1.357	7.03e-3	1.349	3.89e-3
0.410	1.374	1.08e-3	1.342	2.24e-9	1.356	6.99e-3	1.348	3.86e-3
0.440	1.373	1.08e-3	1.340	1.84e-9	1.354	6.94e-3	1.347	3.84e-3
0.470	1.371	1.08e-3	1.339	1.51e-9	1.353	6.89e-3	1.345	3.81e-3
0.510	1.370	1.08e-3	1.337	1.35e-9	1.352	6.87e-3	1.344	3.79e-3
0.550	1.369	1.21e-3	1.336	2.13e-9	1.350	6.76e-3	1.343	3.74e-3
0.610	1.369	1.21e-3	1.335	8.64e-9	1.350	6.66e-3	1.342	3.68e-3
0.670	1.368	1.29e-3	1.334	2.34e-8	1.349	6.64e-3	1.341	3.67e-3
0.750	1.366	1.53e-3	1.333	7.32e-8	1.348	6.72e-3	1.340	3.72e-3
0.865	1.364	2.00e-3	1.332	4.09e-7	1.346	6.87e-3	1.338	3.80e-3
1.040	1.362	2.56e-3	1.329	5.80e-6	1.344	7.18e-3	1.336	3.97e-3
1.240	1.356	2.93e-3	1.326	2.85e-5	1.340	7.43e-3	1.333	4.12e-3
1.640	1.341	3.16e-3	1.318	1.16e-4	1.330	7.86e-3	1.324	4.39e-3
2.250	1.304	2.10e-3	1.295	4.20e-4	1.303	8.17e-3	1.298	4.69e-3

Table 8: Optical properties for the aerosol modes for RH = 98%.